Module Code & Module Title

CS5004NI EMERGING PROGRAMMING PLATFORMS AND TECHNOLOGIES

Assessment Weightage & Type

30% Individual Coursework

Year and Semester

2021 Spring

Student Name: Sanju Sinjapati Magar

London Met ID: 19031620

College ID: np01cp4a190338

Assignment Due Date: 7th May 2021

Assignment Submission Date: 7th May 2021

Title (Where Required): Glaze Music Store

Word Count (Where Required): 2441(From testing to conclusion)

Table of Contents

List of Tables

Table of Figures

1. Introduction

The coursework of this module accounts for 30% of the overall module grade. For this, we must develop a system for a music store named “Glaze Music Store” as an XML developer. So, we are going to make an XML document and for validation and structure of XML, DTD and Schema are used and CSS is used to provide design for the document. XML stands for Extensible Markup Language. It is specified in the World Wide Web Consortium's (W3C) XML 1.0 specification (World Wide Web Consortium). XML offers an easy and standard way to encrypt data and text so that it can be shared across driver hardware, operating systems, and applications with little human interaction. XML is designed in such a way that it can be used to store and transport data that is readable by users.

DTD stands for Document Type Definition which is a document that defines the structure of an XML document. XSD stands for XML Schema Definition which is a way to describe the structure of an XML document. CSS stands for Cascading Style Sheets which can be used to add style and display information to an XML document and can format the entire XML document.

We have to do proper testing of the documents of XML, DTD, and Schema along with CSS with no errors. For this validation needs to be done with the help of the validation site like XML validation which enables us to check the errors occurred in XML, DTD and Schema document and solve the errors according to it.

2. Tree Diagram

A tree diagram is a new management strategy method that shows the hierarchy of tasks and subtasks required to achieve a goal. The tree diagram begins with a single item that divides into two or more, each of which divides into two or more, and so on. With a trunk and several branches, the finished diagram resembles a tree. (ASQ, 2021)

The tree diagram was made with the help of draw.io. Draw.io is a free online diagram software where we can create different kinds of model diagrams that are free to use which I mainly use this tool to draw different kinds of models and diagrams

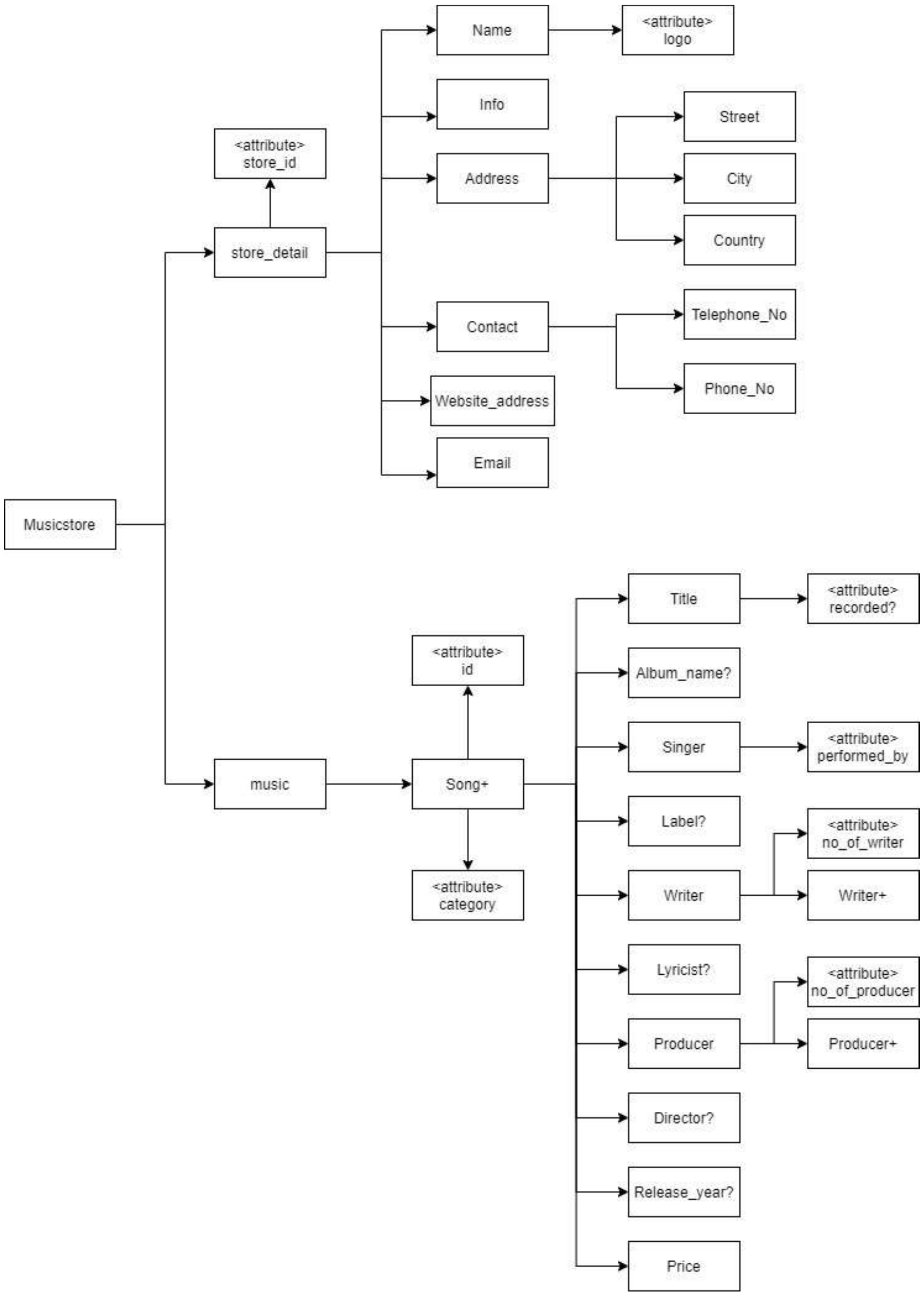


Figure 1: Tree diagram

3. XML Document

```
 <?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="catalog_19031620.css"?>
<!-- Linking the CSS file-->
<!DOCTYPE Musicstore SYSTEM "catalog_19031620.dtd">
<!-- Linking external DTD file-->
<Musicstore xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
xs:noNamespaceSchemaLocation="catalog_19031620.xsd">
<!-- Linking the Schema file-->
<!-- Making Musicstore root element -->
   <store_detail store_id="0281">
   <!-- Defining child element of Musicstore -->
      <Name logo="Glaze">Glaze Music Store</Name>
      <Info>
      <![CDATA[
      Welcome to the Glaze's Music Store. We provide CDs and DVDs of songs of different Genr
es like Pop, Jazz, Rock, Country Music & Dubstep in a affortable price.
      ]]>
      <!-- Defining character data -->
      </Info>
      <Address>
         <Street>Location: New Road,</Street>
         <City>Kathmandu,</City>
         <Country>Nepal</Country>
      </Address>
      <Contact>
      <Telephone_No>Contact: 071-544281,</Telephone_No>
      <Phone_No>9847823223</Phone_No>
      </Contact>
      <Website_address>Visit: www.GlazeMusicStore.com</Website_address>
      <Email>Email: GlazeMusic@gmail.com</Email>
   </store_detail>

   <music>
   <!-- Defining child element of Musicstore -->
      <Song category="Jazz" id="JQT1">
      <!-- Defining child element of music with two attributes-->
         <Title>Quick Trick</Title>
         <Album_name>Album: Just Coolin</Album_name>
         <Singer performed_by="Band">By: Art Blakery &The Jazz Messengers</Singer>Writt
en By:
         <!-- defining character reference of & -->
         <Writers no_of_writer="1">
            <Writer>Bobby Timmons</Writer>
         </Writers>Produced By:
         <Producers no_of_producer="2">
```

```
            <Producer>Alfred Lion</Producer>
            <Producer>Zev Feldman</Producer>
        </Producers>Release Year:
        <Release_year> 2020</Release_year>
        <Price>Price: Rs200</Price>
    </Song>
    <Song category="Jazz" id="JBM2">
        <Title>Blue Moon</Title>
        <Singer performed_by="Solo">By: Billie Holiday</Singer>Written By:
        <Writers no_of_writer="2">
            <Writer>Billie Holiday</Writer>
            <Writer>Richard Rodgers</Writer>
        </Writers>
        <Lyricist>Lyricist: Lorenz Hertz</Lyricist>Produced By:
        <Producers no_of_producer="1">
            <Producer>Norman Granz</Producer>
        </Producers>Release Year:
        <Release_year> 1934</Release_year>
        <Price>Price: Rs200</Price>
    </Song>
    <Song category="Rock" id="RST3">
        <Title>Smells Like Teen Spirit</Title>
        <Album_name>Album: Nevermind (Deluxe Edition)</Album_name>
        <Singer performed_by="Band">By: Nirvana</Singer>
        <Label>Label: DGC</Label>Written By:
        <Writers no_of_writer="3">
            <Writer>Dave Grohl</Writer>
            <Writer>Krist Noveoselic</Writer>
            <Writer>Kurt Cobain</Writer>
        </Writers>Produced By:
        <Producers no_of_producer="1">
            <Producer>Butch Vig</Producer>
        </Producers>Release Year:
        <Release_year> 2020</Release_year>
        <Price>Price: Rs300</Price>
    </Song>
    <Song category="Rock" id="RWJ4">
        <Title>Welcome to the Jungle</Title>
        <Album_name>Album: Appetite for Destruction</Album_name>
        <Singer performed_by="Band">By: Guns N' Roses</Singer>
        <!-- defining character reference of " ' " -->
        <Label>Label: Geffen</Label>Written By:
        <Writers no_of_writer="1">
            <Writer>Guns N' Roses</Writer>
        </Writers>Produced By:
        <Producers no_of_producer="1">
```

```
            <Producer>Mike Clink</Producer>
        </Producers>Release year:
        <Release_year>1987</Release_year>
        <Price>Price: Rs320</Price>
    </Song>
    <Song category="Rock" id="RCS5">
        <Title recorded="Cello Studios">Chop Suey</Title>
        <Album_name>Album: Toxicity</Album_name>
        <Singer performed_by="Band">By: System of A Down</Singer>
        <Label>Label: Geffen</Label>Written By:
        <Writers no_of_writer="2">
            <Writer>Daron Malakian</Writer>
            <Writer>Serj Tankian</Writer>
        </Writers>Produced By:
        <Producers no_of_producer="2">
            <Producer>Rick Rubin</Producer>
            <Producer>Daron Malakian</Producer>
        </Producers>Release year:
        <Release_year>2001</Release_year>
        <Price>Price: Rs250</Price>
    </Song>
    <Song category="Rock" id="RBD6">
        <Title>Boulevard of Broken Dreams</Title>
        <Album_name>Album: American Idiot</Album_name>
        <Singer performed_by="Band">By: Green Day</Singer>
        <Label>Label: Reprise, Warner Bros.</Label>Written By:
        <Writers no_of_writer="3">
            <Writer>Billie Joe Armstrong</Writer>
            <Writer>Mike Dirnt</Writer>
            <Writer>Tre Cool</Writer>
        </Writers>Produced By:
        <Producers no_of_producer="2">
            <Producer>Rob Cavallo</Producer>
            <Producer>Green Day</Producer>
        </Producers>
        <Price>Price: Rs320</Price>
    </Song>
    <Song category="Country" id="CBH7">
        <Title>Beer Never Broke My Heart</Title>
        <Album_name>Album: What You See Is What You Get</Album_name>
        <Singer performed_by="Band">By: Luke Combs</Singer>
        <Label>Label: Columbia Nashville</Label>Written By:
        <Writers no_of_writer="3">
            <Writer>Luke Combs</Writer>
            <Writer>Randy Montana</Writer>
            <Writer>Jonathan Singleton</Writer>
        </Writers>Produced By:
        <Producers no_of_producer="1">
            <Producer>Scott Moffatt</Producer>
        </Producers>Release Year:
        <Release_year> 2019</Release_year>
        <Price>Price: Rs180</Price>
    </Song>
    <Song category="Country" id="CGC8">
        <Title recorded="American country music">God's Country</Title>
        <Album_name>Album: God's Country</Album_name>
        <Singer performed_by="Solo">By: Blake Shelton</Singer>
        <Label>Label: Warner Bros. Nashville</Label>Written By:
        <Writers no_of_writer="3">
            <Writer>Devin Dawson</Writer>
            <Writer>Jordan Schmidt</Writer>
            <Writer>Hardy</Writer>
        </Writers>Produced By:
        <Producers no_of_producer="1">
            <Producer>Scott Hendricks</Producer>
        </Producers>Release Year:
        <Release_year> 2019</Release_year>
        <Price>Price: Rs180</Price>
    </Song>
    <Song category="Country" id="CPA9" >
        <Title recorded="American country music">Play It Again</Title>
        <Album_name>Album: Crash My Party</Album_name>
        <Singer performed_by="Solo">By: Luke Bryan</Singer>
        <Label>Label: Capitol Nashville</Label>Written By:
        <Writers no_of_writer="2">
            <Writer>Dallas Davidson</Writer>
            <Writer>Ashley Gorley</Writer>
        </Writers>Produced By:
        <Producers no_of_producer="1">
            <Producer>Jeff Stevens</Producer>
        </Producers>Release Year:
        <Release_year> 2014</Release_year>
        <Price>Price: Rs200</Price>
    </Song>
    <Song category="Pop" id="PBN10">
        <Title>Better Now</Title>
        <Album_name>Album: Beerbongs & Bentleys</Album_name>
        <Singer performed_by="Solo">By: Post Malone</Singer>
        <Label>Label: Republic</Label>Written By:
        <Writers no_of_writer="5">
            <Writer>Post Malone</Writer>
            <Writer>Louis Bell</Writer>
        <Writer>Frank Dukes</Writer>
        <Writer>Billy Walsh</Writer>
        <Writer>Kaan Gunesberk</Writer>
      </Writers>Produced By:
      <Producers no_of_producer="2">
        <Producer>Alfred Lion</Producer>
        <Producer>Louis Bell</Producer>
      </Producers>Release Year:
      <Release_year> 2018</Release_year>
      <Price>Price: Rs280</Price>
    </Song>
    <Song category="Pop" id="PA11">
      <Title>Attention</Title>
      <Album_name>Album: Voicenotes</Album_name>
      <Singer performed_by="Solo">By: Charlie Puth</Singer>
      <Label>Label: Artist Partner, Atlantic</Label>Written By:
      <Writers no_of_writer="2">
        <Writer>Charlie Puth</Writer>
        <Writer>Jacob Kasher</Writer>
      </Writers>Produced By:
      <Producers no_of_producer="1">
        <Producer>Charlie Puth</Producer>
      </Producers>Release Year:
      <Release_year> 2017</Release_year>
      <Price>Price: Rs250</Price>
    </Song>
    <Song category="Pop" id="PDD12">
      <Title recorded="RCA records">Dusk Till Dawn</Title>
      <Album_name>Album: Icarus Falls</Album_name>
      <Singer performed_by="Duo">By: Zayn Malik &Sia</Singer>
      <Label>Label: Republic</Label>Written By:
      <Writers no_of_writer="5">
        <Writer>Zayn Malik</Writer>
        <Writer>Sia Furler</Writer>
        <Writer>Alex Oriet</Writer>
        <Writer>David Phelan</Writer>
        <Writer>Greg Kustin</Writer>
      </Writers>Produced By:
      <Producers no_of_producer="1">
        <Producer>Greg Kustin</Producer>
      </Producers>
      <Director>Director: Marc Webb</Director>Release Year:
      <Release_year> 2018</Release_year>
      <Price>Price: Rs280</Price>
    </Song>
    <Song category="Pop" id="PP13">
```

```
        <Title>Perfect</Title>
        <Singer performed_by="Solo">By: Ed Shereen</Singer>
        <Label>Label: Asylum, Atlantic</Label>Written By:
        <Writers no_of_writer="1">
          <Writer>Ed Sheeran</Writer>
        </Writers>Produced By:
        <Producers no_of_producer="2">
          <Producer>Ed Sheeran</Producer>
          <Producer>Will Hicks</Producer>
        </Producers>
        <Director>Director: Jason Konin</Director>Release Year:
        <Release_year> 2017</Release_year>
        <Price>Price: Rs300</Price>
      </Song>
      <Song category="Dubstep" id="DBB14">
        <Title>Bunker Buster</Title>
        <Album_name>Album: Bunker Buster</Album_name>
        <Singer performed_by="Duo">By: Excision & Subtronics</Singer>Written By:
        <Writers no_of_writer="2">
          <Writer>Excision</Writer>
          <Writer>Subtronics</Writer>
        </Writers>Produced By:
        <Producers no_of_producer="2">
          <Producer>Alfred Lion</Producer>
          <Producer>Louis Bell</Producer>
        </Producers>Release Year:
        <Release_year>2021</Release_year>
        <Price>Price: Rs150</Price>
      </Song>
      <Song category="Dubstep" id="DA15">
        <Title>Alone</Title>
        <Singer performed_by="Solo">By: Marshmello</Singer>
        <Label>Label: Monster Cat</Label>Written By:
        <Writers no_of_writer="1">
          <Writer>Marshmello</Writer>
        </Writers>Produced By:
        <Producers no_of_producer="2">
          <Producer>Karam Gill</Producer>
          <Producer>Jack Winter</Producer>
        </Producers>
        <Director>Director: Daniel Burke</Director>Year:
        <Release_year>2016</Release_year>
        <Price>Price: Rs220</Price>
      </Song>
    </music>
</Musicstore>
```

```
<!-- Ending the root element -->
```

4. DTD Content

```
  <!ELEMENT Musicstore (store_detail, music)>
<!--Declaring the root element first-->
<!ATTLIST Musicstore xmlns:xs CDATA #REQUIRED>
<!ATTLIST Musicstore xs:noNamespaceSchemaLocation CDATA #REQUIRED>
<!--Declaring the attributes of namespace xs which contains the XSD location-->
<!ELEMENT store_detail (Name,Info,Address,Contact,Website_address,Email)>
<!--Declaring store_detail element with 6 child elements-->
<!ATTLIST store_detail store_id CDATA #REQUIRED>
<!--
Declaring the attribute of store_detail store_id which has character data and it mandatory to defin
e it-->
<!ATTLIST Name logo CDATA #REQUIRED>
<!ELEMENT Name (#PCDATA)>
<!--Declaring parsed character data in element Name-->
<!ELEMENT Info (#PCDATA)>
<!ELEMENT Address (Street,City,Country)>
<!--Declaring Address element with 3 child elements-->
<!ELEMENT Street (#PCDATA)>
<!ELEMENT City (#PCDATA)>
<!ELEMENT Country (#PCDATA)>
<!ELEMENT Contact (Telephone_No,Phone_No)>
<!--Declaring Contact element with 2 child elements-->
<!ELEMENT Telephone_No (#PCDATA)>
<!ELEMENT Phone_No (#PCDATA)>
<!ELEMENT Website_address (#PCDATA)>
<!ELEMENT Email (#PCDATA)>
<!ELEMENT music (Song+)>
<!ELEMENT Song (#PCDATA|Title|Album_name|Singer|Label|Writers|Lyricist|Producers|Dire
ctor|Release_year|Price)*>
<!--
Declaring the mixed content where the Song element can contain zero or more occurrence of par
sed character of data and other 10 elements-->
<!ATTLIST Song category (Jazz | Pop | Rock | Country | Dubstep) #REQUIRED>
<!--
Declaring the attribute category of Song element which can either have any 5 of the given values
 and is mandatory to include any one of them-->
<!ATTLIST Song id ID #REQUIRED>
<!--Declaring the attribute id which must have an ID type in Song element-->
<!ELEMENT Title (#PCDATA)>
```

```
<!ATTLIST Title recorded CDATA #IMPLIED>
<!--
Declaring the attribute recorded which can have character data and it is optional to define the attr
ibute Title-->
<!ELEMENT Album_name (#PCDATA)>
<!ELEMENT Singer (#PCDATA)>
<!ATTLIST Singer performed_by CDATA #REQUIRED>
<!ELEMENT Label (#PCDATA)>
<!ELEMENT Writers (Writer+)>
<!--Declaring the element Writers where it must contain 1 or multiple child element Writer-->
<!ATTLIST Writers no_of_writer CDATA #REQUIRED>
<!ELEMENT Writer (#PCDATA)>
<!ELEMENT Lyricist (#PCDATA)>
<!ELEMENT Producers (Producer+)>
<!--Declaring the element Producers where it must contain 1 or multiple child element Producer-
->
<!ATTLIST Producers no_of_producer CDATA #REQUIRED>
<!ELEMENT Producer (#PCDATA)>
<!ELEMENT Director (#PCDATA)>
<!ELEMENT Release_year (#PCDATA)>
<!ELEMENT Price (#PCDATA)>
```

5. Schema Content

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Musicstore">
  <!-- Declaring root element with two child elements -->
     <xs:complexType><!-- Declaring complex type as it has child elements-->
        <xs:sequence>
           <xs:element ref="store_detail" maxOccurs="1"/>
           <xs:element ref="music" maxOccurs="1"/>
        </xs:sequence>
     </xs:complexType>
  </xs:element>
  <xs:element name="store_detail">
  <!-- Declaring the store-detail element-->
     <xs:complexType><!-- Declaring complex type as it has child elements and one attribute-->
        <xs:sequence>
           <xs:element ref="Name"/><!-- Referencing the elements from the below -->
           <xs:element ref="Info"/>
           <xs:element ref="Address"/>
           <xs:element ref="Contact"/>
```

```
            <xs:element ref="Website_address"/>
            <xs:element ref="Email"/>
         </xs:sequence>
         <xs:attribute name="store_id" type="xs:positiveInteger"/><!-
- Defining the positiveInteger in the attribute -->
      </xs:complexType>
   </xs:element>
   <xs:element name="Name">
      <xs:complexType><!-- Declaring complex type as it has attribute-->
         <xs:simpleContent><!-
- Defining simpleContent as it has no child elements but only PCDATA-->
            <xs:extension base="xs:string">
               <xs:attribute name="logo" type="xs:string" use="required"/>
            </xs:extension>
         </xs:simpleContent>
      </xs:complexType>
   </xs:element>
   <xs:element name="Info" type="xs:string"/><!-- Defining simple type element-->
   <xs:element name="Address">
      <xs:complexType><!-- Declaring complex type as it has three child elements-->
         <xs:sequence>
            <xs:element name="Street" type="xs:string"/>
            <xs:element name="City" type="xs:string"/>
            <xs:element name="Country" type="xs:string"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="Contact">
      <xs:complexType><!-- Declaring complex type as it has child elements-->
         <xs:sequence>
            <xs:element ref="Telephone_No"/>
            <xs:element ref="Phone_No"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="Telephone_No" type="xs:string"/><!-- Defining simple type element-->
   <xs:element name="Phone_No">
      <xs:simpleType><!-
- Defining simple type element where certain restriction and pattern are set fro phone number-->
            <xs:restriction base="xs:positiveInteger">
               <xs:pattern value="98[0-9]{8}"/>
            </xs:restriction>
      </xs:simpleType>
   </xs:element>
   <xs:element name="Website_address" type="xs:anyURI"/><!-
- Defining simple type element-->
```

```
  <xs:element name="Email"><!-
- Defining simple type element where certain restriction and pattern are set for email-->
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:pattern value="Email: ([a-z]|[A-Z])+[@]gmail[.]com"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="music"><!-- Declaring complex type as it has child element-->
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Song" minOccurs="15" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Song"><!-
- Declaring mixed complex type as Song element contains PCDATA along with child elements a
nd attributes-->
    <xs:complexType mixed="true">
      <xs:sequence>
        <xs:element ref="Title"/>
        <xs:element ref="Album_name" minOccurs="0" maxOccurs="1"/><!-
- Defining element as optional -->
        <xs:element ref="Singer"/>
        <xs:element ref="Label" minOccurs="0" maxOccurs="1"/><!-
- Defining element as optional -->
        <xs:element ref="Writers"/>
        <xs:element ref="Lyricist" minOccurs="0" maxOccurs="1"/><!-
- Defining element as optional -->
        <xs:element ref="Producers"/>
        <xs:element ref="Director" minOccurs="0" maxOccurs="1"/><!-
- Defining element as optional -->
        <xs:element ref="Release_year" minOccurs="0" maxOccurs="1"/><!-
- Defining element as optional -->
        <xs:element ref="Price"/>
      </xs:sequence>
        <xs:attribute ref="category" use="required"/>
        <xs:attribute name="id" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="Title"><!-- Declaring element complex type as it includes attribute-->
    <xs:complexType>
      <xs:simpleContent><!-
- Defining simple content as it contains only PCDATA with no child element -->
        <xs:extension base="xs:string">
```

```
                <xs:attribute name="recorded" type="xs:string" use="optional"/><!-
- Attribute is set to optional-->
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="Album_name" type="xs:string"/><!-- Declaring element simple type -->
  <xs:element name="Singer"><!-- Declaring element complex type as it includes attribute-->
    <xs:complexType>
        <xs:simpleContent><!-
- Defining simple content as it contains only PCDATA with no child element -->
            <xs:extension base="xs:string">
                <xs:attribute ref="performed_by" use="required"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="Label" type="xs:string"/><!-- Declaring element simple type -->
  <xs:element name="Writers"><!-
- Declaring element complex type as it includes child elements with attribute-->
    <xs:complexType>
        <xs:sequence>
            <xs:element name="Writer" minOccurs="1" maxOccurs="unbounded"/><!-
- Defining the element Writer which must occur atleast one time or multiple times-->
        </xs:sequence>
        <xs:attribute ref="no_of_writer" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="Lyricist" type="xs:string"/><!-- Declaring element simple type -->
  <xs:element name="Producers"><!-
- Declaring element complex type as it includes child elements with attribute-->
    <xs:complexType>
        <xs:sequence>
            <xs:element name="Producer" minOccurs="1" maxOccurs="unbounded"/><!-
- Defining the element Writer which must occur atleast one time or multiple times-->
        </xs:sequence>
        <xs:attribute ref="no_of_producer" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="Director" type="xs:string"/><!-- Declaring element simple type -->
  <xs:element name="Release_year" type="xs:string"/><!-- Declaring element simple type -->
  <xs:element name="Price" type="xs:string"/><!-- Declaring element simple type -->
  <xs:attribute name="category">
    <xs:simpleType>
        <xs:restriction base="xs:string"><!-
- Defining the attribute with restriction of string data type -->
```

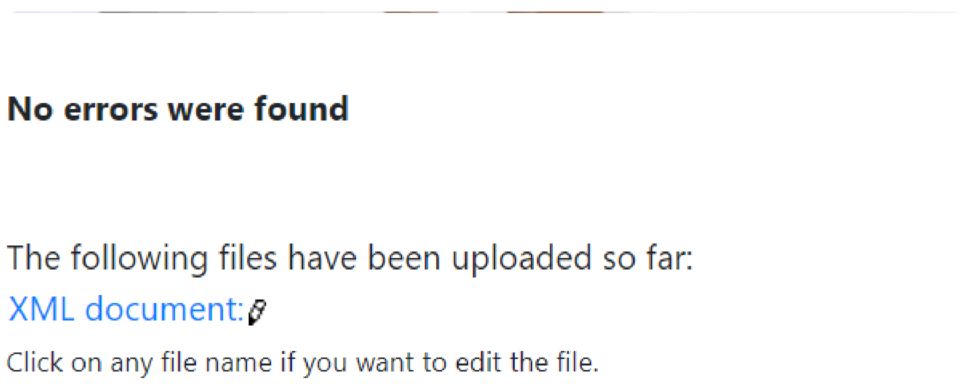
```
          <!-
- Setting  the attribute with enumeration values where any one values among them should be the value of the attribute -->
            <xs:enumeration value="Jazz"/>
            <xs:enumeration value="Country"/>
            <xs:enumeration value="Rock"/>
            <xs:enumeration value="Pop"/>
            <xs:enumeration value="Dubstep"/>
          </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="performed_by">
        <xs:simpleType>
          <xs:restriction base="xs:string"><!-
- Setting the restriction on attribute of string data type -->
            <xs:pattern value="Solo|Duo|Band"/><!-
- Setting the pattern on attribute which could contain any one values among them -->
          </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="no_of_writer">
        <xs:simpleType>
          <xs:restriction base="xs:positiveInteger"><!-
- Setting the restriction on attribute of positive integer data type -->
            <xs:pattern value="[0-9]"/><!-
- Setting the pattern on attribute which could contain any number from 0 to 9 -->
          </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="no_of_producer">
        <xs:simpleType>
          <xs:restriction base="xs:positiveInteger"><!-
- Setting the restriction on attribute of positive integer data type -->
            <xs:pattern value="[0-9]"/><!-
- Setting the pattern on attribute which could contain any number from 0 to 9 -->
          </xs:restriction>
        </xs:simpleType>
    </xs:attribute>
</xs:schema>
```

6. Testing

Table 1: Test 1

Test 1	Checking XML well-formedness.
Action	Pasting the code of XML on XML validation site.
Expected Output	No error is expected to be found.
Actual Output	No error was found.
Result	Successful

Screenshot:



No errors were found

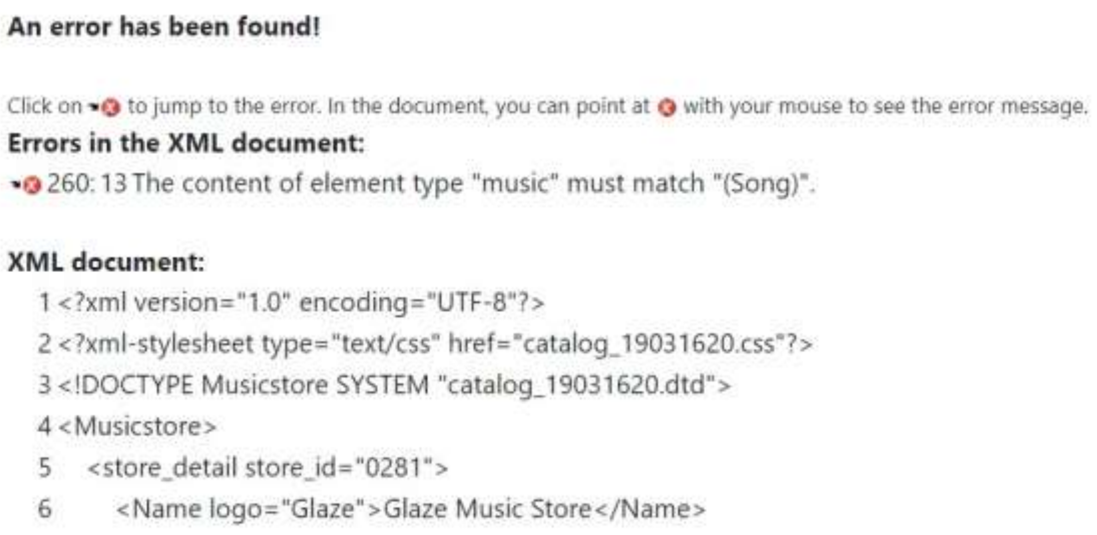
The following files have been uploaded so far:
XML document:
Click on any file name if you want to edit the file.

Figure 2: SS of XML validation

Table 2: Test 2

Test 2	XML and DTD validation.
Action	Pasting the code of XML and DTD on XML validation site.
Expected Output	No error is expected to be found.
Actual Output	Error was found
Result	Unsuccessful

Screenshot:



An error has been found!

Click on ▾⊗ to jump to the error. In the document, you can point at ⊗ with your mouse to see the error message.

Errors in the XML document:

▾⊗ 260: 13 The content of element type "music" must match "(Song)".

XML document:

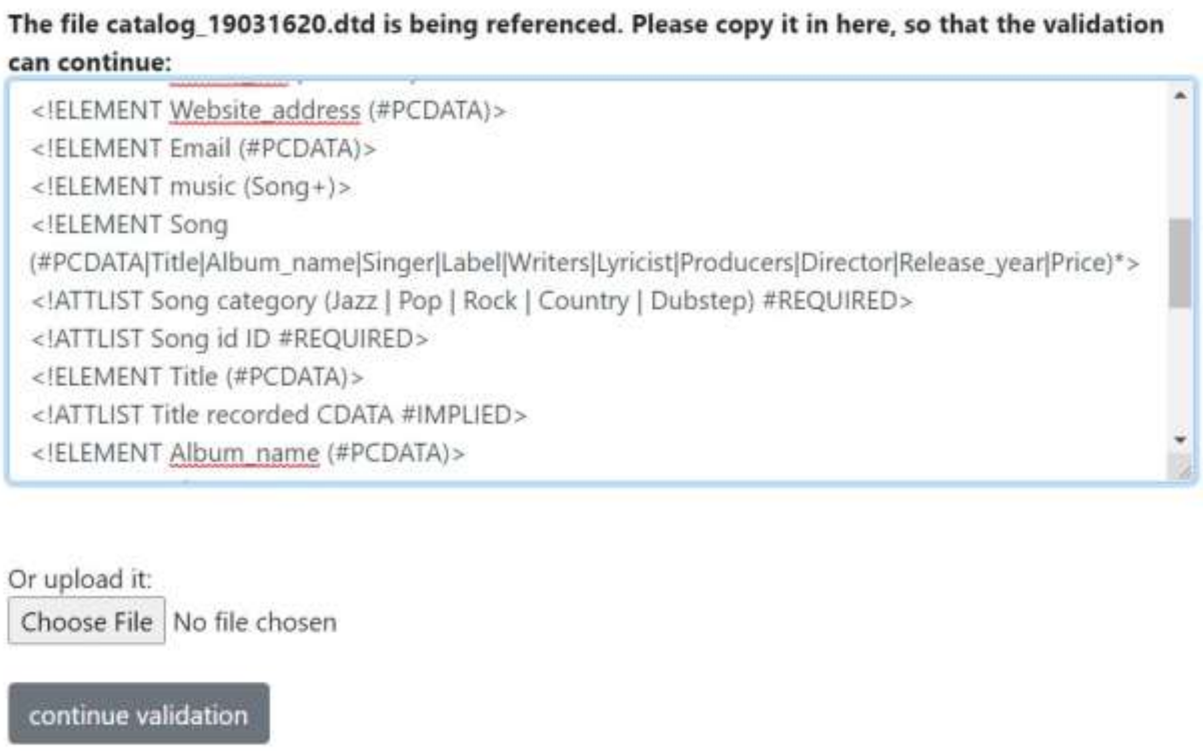
```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="catalog_19031620.css"?>
<!DOCTYPE Musicstore SYSTEM "catalog_19031620.dtd">
<Musicstore>
    <store_detail store_id="0281">
        <Name logo="Glaze">Glaze Music Store</Name>
```

Figure 3: SS of error in DTD

Table 3: Test 3

Test 3	XML and DTD validation.
Action	Debugging the error faced in Song element in DTD and again validating on XML validation site.
Expected Output	No error is expected to be found.
Actual Output	No error was found.
Result	Successful

Screenshot:

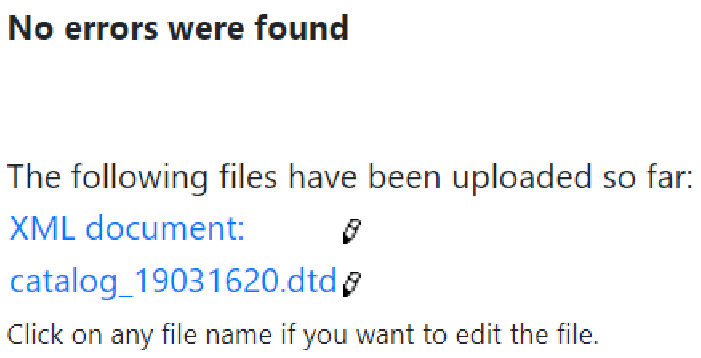


The file catalog_19031620.dtd is being referenced. Please copy it in here, so that the validation can continue:

```
<!ELEMENT Website_address (#PCDATA)>
<!ELEMENT Email (#PCDATA)>
<!ELEMENT music (Song+)>
<!ELEMENT Song
(#PCDATA|Title|Album_name|Singer|Label|Writers|Lyricist|Producers|Director|Release_year|Price)*>
<!ATTLIST Song category (Jazz | Pop | Rock | Country | Dubstep) #REQUIRED>
<!ATTLIST Song id ID #REQUIRED>
<!ELEMENT Title (#PCDATA)>
<!ATTLIST Title recorded CDATA #IMPLIED>
<!ELEMENT Album_name (#PCDATA)>
```

Or upload it:

Figure 4: SS Of Debugging DTD



No errors were found

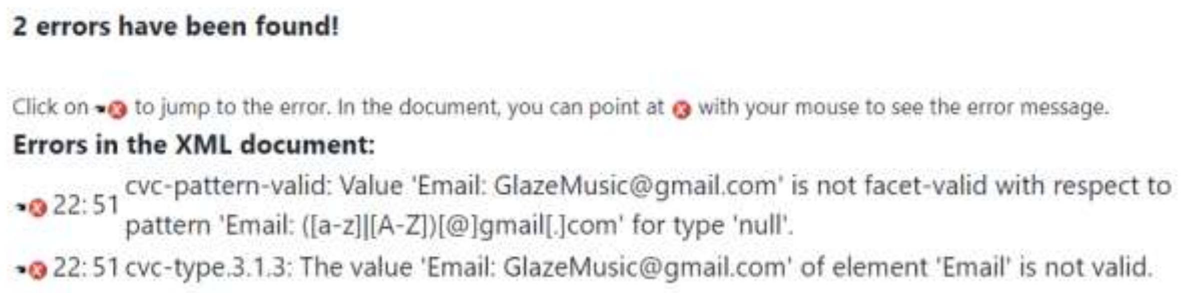
The following files have been uploaded so far:
XML document:
catalog_19031620.dtd
Click on any file name if you want to edit the file.

Figure 5: SS of Error-free in DTD and XML

Table 4: Test 4

Test 4	XML and XSD validation.
Action	Pasting the code of XML and XSD on XML validation site.
Expected Output	No error is expected to be found.
Actual Output	2 errors were found.
Result	Unsuccessful

Screenshot:



2 errors have been found!

Click on to jump to the error. In the document, you can point at with your mouse to see the error message.

Errors in the XML document:

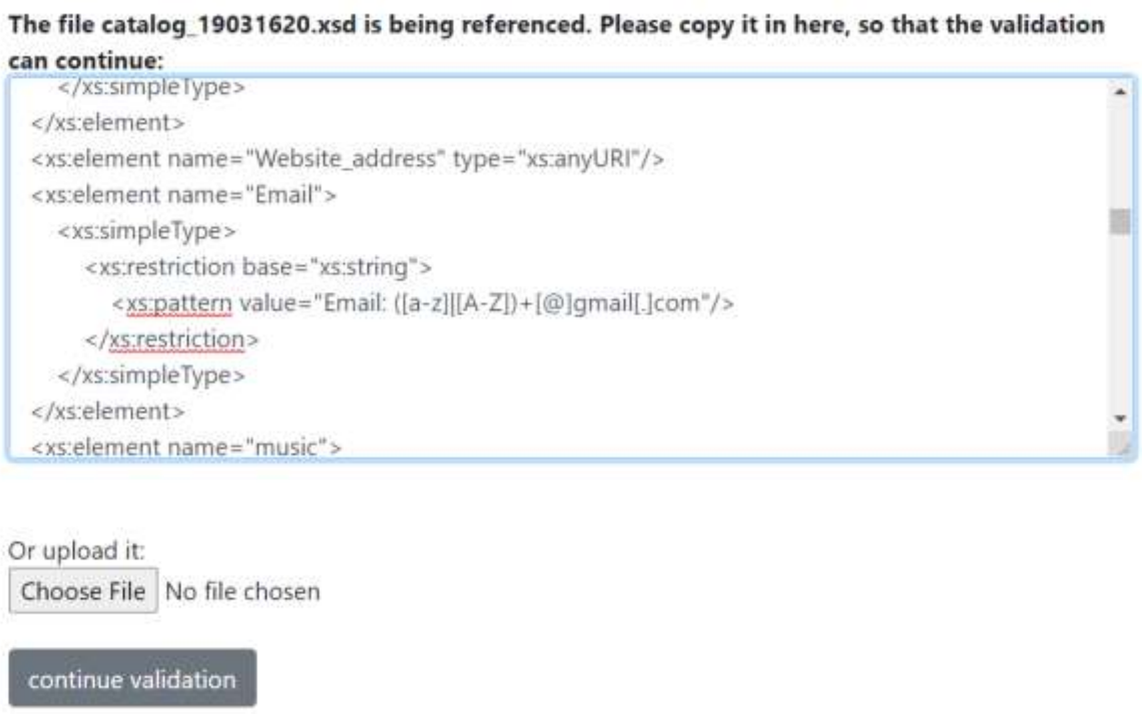
```
22:51 cvc-pattern-valid: Value 'Email: GlazeMusic@gmail.com' is not facet-valid with respect to
      pattern 'Email: ([a-z]|[A-Z])[@]gmail[.]com' for type 'null'.
22:51 cvc-type.3.1.3: The value 'Email: GlazeMusic@gmail.com' of element 'Email' is not valid.
```

Figure 6: SS of error in XSD

Table 5: Test 5

Test 5	XML and XSD validation.
Action	Debugging pattern of Email in XSD where "+" was missing after the ([a-z]|[A-Z]) and again validating on XML validation site.
Expected Output	No error is expected to be found.
Actual Output	No error was found.
Result	Successful

Screenshot:



The file catalog_19031620.xsd is being referenced. Please copy it in here, so that the validation can continue:

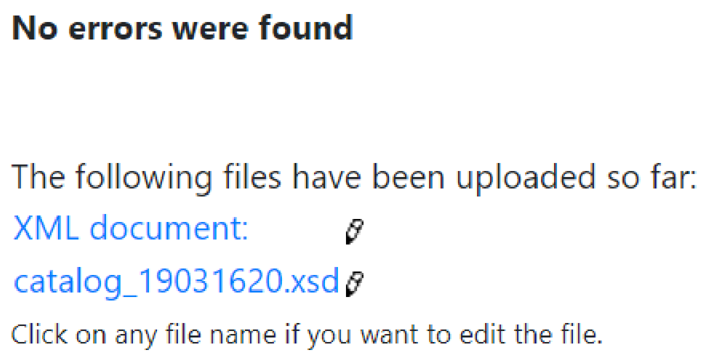
```
    </xs:simpleType>
  </xs:element>
  <xs:element name="Website_address" type="xs:anyURI"/>
  <xs:element name="Email">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:pattern value="Email: ([a-z][A-Z])+[@]gmail[.]com"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="music">
```

Or upload it:

Choose File No file chosen

continue validation

Figure 7: SS of Debugging XSD document



No errors were found

The following files have been uploaded so far:

XML document:

catalog_19031620.xsd

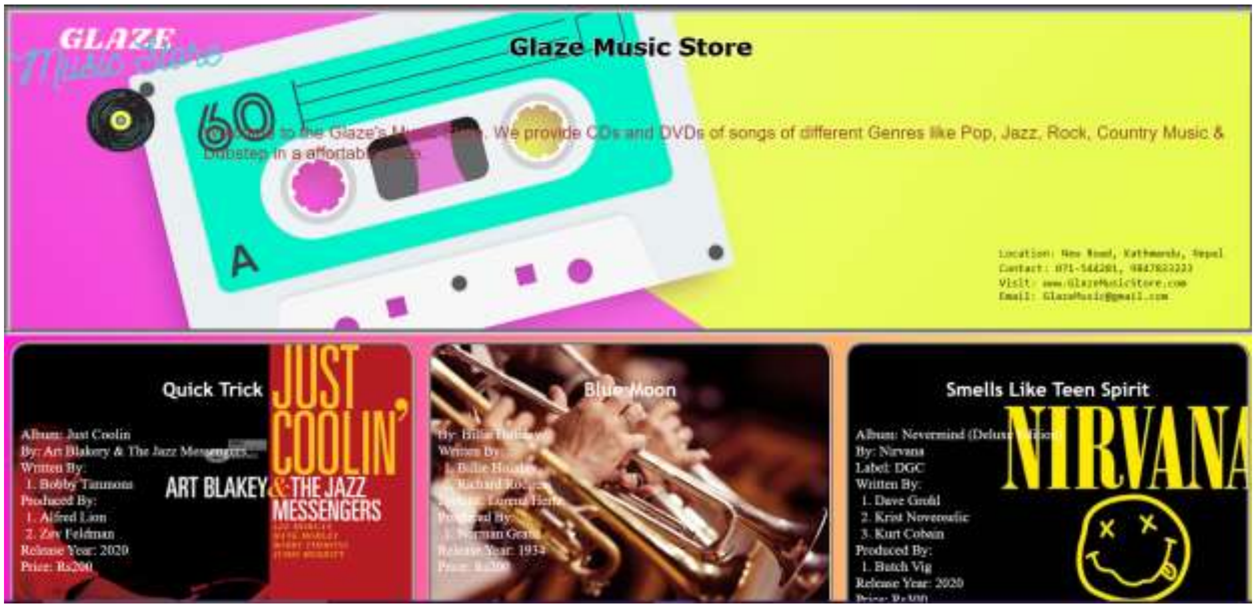
Click on any file name if you want to edit the file.

Figure 8: SS of Error-free in XSD and XML

Table 6: Test 6

Test 6	Displaying CSS styling in browser
Action	Linking CSS in XML.
Expected Output	The XML document is expected to be styled and display information with the help of CSS.
Actual Output	The XML document was styled and displayed information with the help of CSS.
Result	Successful

Screenshot:



Glaze Music Store

[illegible] to the Glaze's [illegible]. We provide CDs and DVDs of songs of different Genres like Pop, Jazz, Rock, Country Music & Dubstep in a affordable [illegible]

Location: New Road, Kathmandu, Nepal
Contact: 071-544201, 9847833333
Visit: www.GlazeMusicStore.com
Email: GlazeMusic@gmail.com

Quick Trick

Album: Just Coolin
By: Art Blakery & The Jazz Messengers
Written By:
1. Bobby Timmons
Produced By:
1. Alfred Lion
2. Zev Feldman
Release Year: 2020
Price: Rs200

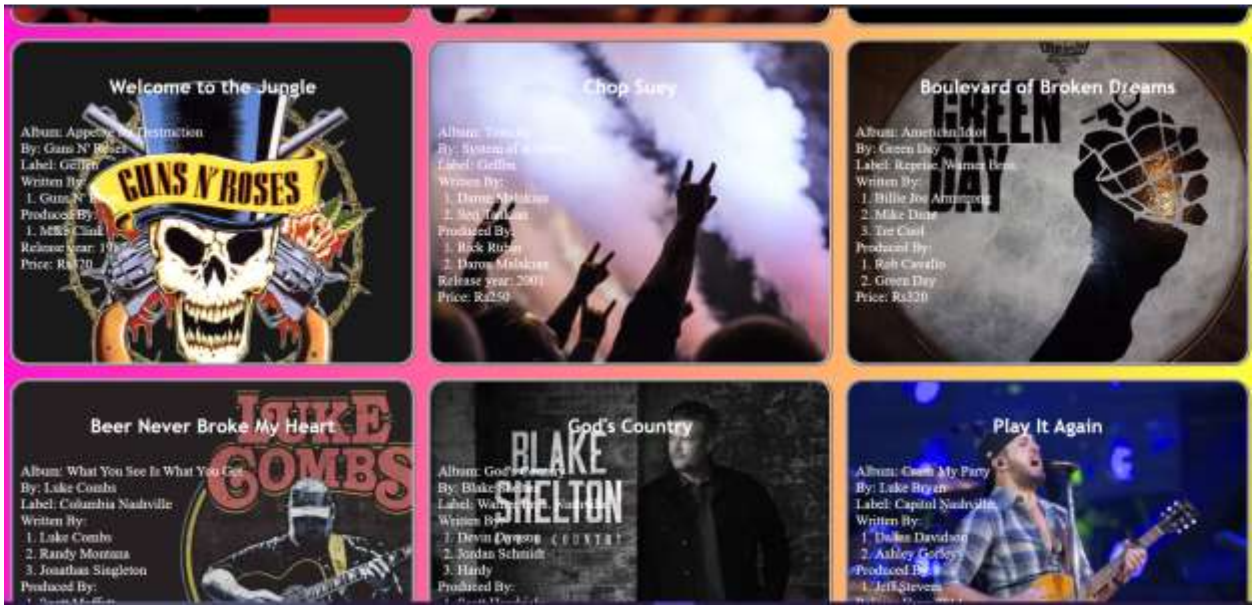
Blue Moon

By: Billie Holiday
Written By:
1. Billie Holiday
2. Richard Rodgers
[illegible]: Lorenz Hart
Produced By:
1. Norman Granz
Release Year: 1934
Price: Rs200

Smells Like Teen Spirit

Album: Nevermind (Deluxe Edition)
By: Nirvana
Label: DGC
Written By:
1. Dave Grohl
2. Krist Novoselic
3. Kurt Cobain
Produced By:
1. Butch Vig
Release Year: 2020
Price: Rs300

Figure 9: CSS styling1



Welcome to the Jungle

Album: Appetite for Destruction
By: Guns N' Roses
Label: Geffen
Written By:
1. Guns N' [illegible]
Produced By:
1. Mike Clink
Release year: 19[illegible]
Price: Rs170

Chop Suey

Album: [illegible]
By: System of a [illegible]
Label: Geffen
Written By:
1. Daron Malakian
2. Serj Tankian
Produced By:
1. Rick Rubin
2. Daron Malakian
Release year: 2001
Price: Rs250

Boulevard of Broken Dreams

Album: American Idiot
By: Green Day
Label: Reprise, Warner Bros
Written By:
1. Billie Joe Armstrong
2. Mike Dirnt
3. Tre Cool
Produced By:
1. Rob Cavallo
2. Green Day
Price: Rs320

Beer Never Broke My Heart

Album: What You See Is What You Get
By: Luke Combs
Label: Columbia Nashville
Written By:
1. Luke Combs
2. Randy Montana
3. Jonathan Singleton
Produced By:

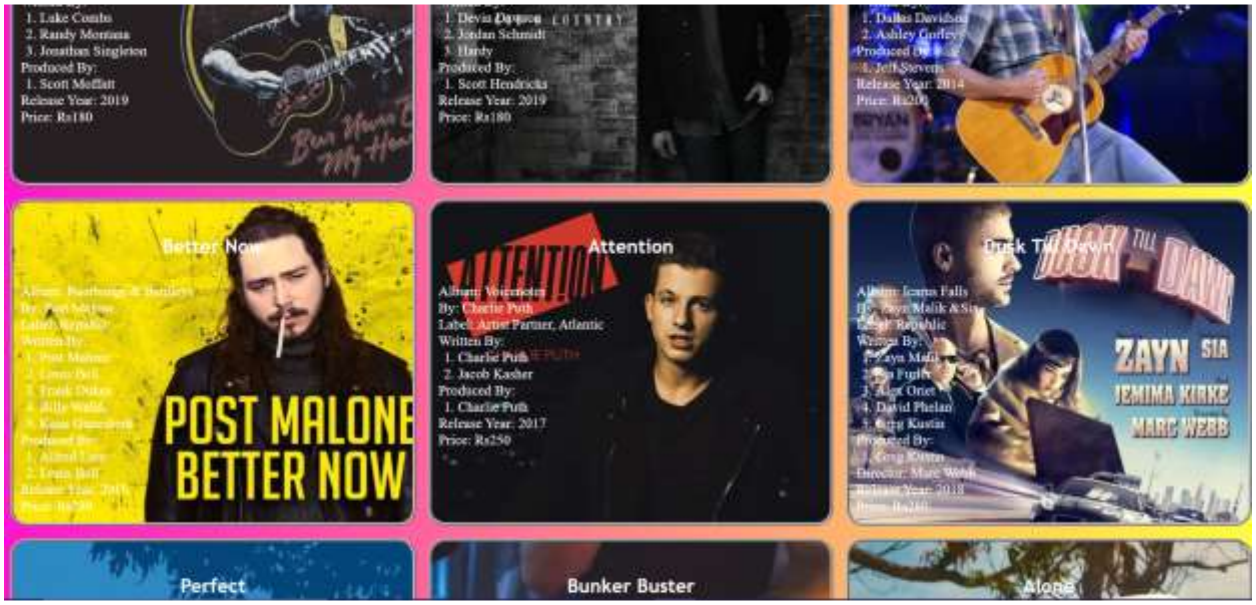
God's Country

Album: God's Country
By: Blake Shelton
Label: Wa[illegible]
Written By:
1. Devin Dawson
2. Jordan Schmidt
3. Hardy
Produced By:

Play It Again

Album: Crash My Party
By: Luke Bryan
Label: Capitol Nashville
Written By:
1. Dallas Davidson
2. Ashley Gorley
Produced By:
1. Jeff Stevens

Figure 10: CSS styling2



1. Luke Combs
2. Randy Montana
3. Jonathan Singleton
Produced By:
1. Scott Moffatt
Release Year: 2019
Price: Rs180

1. Devin Dawson
2. Jordan Schmidt
3. Hardy
Produced By:
1. Scott Hendricks
Release Year: 2019
Price: Rs180

1. Dallas Davidson
2. Ashley Gorley
Produced By:
1. Jeff Stevens
Release Year: 2014
Price: Rs200

Better Now

Album: Beerbongs & Bentleys
By: Post Malone
Label: Republic
Written By:
1. Post Malone
2. Louis Bell
3. Frank Dukes
4. Billy Walsh
5. Kaan Gunesberk
Produced By:
1. Alfred Lion
2. Louis Bell
Release Year: 2018
Price: Rs200

Attention

Album: Voicenotes
By: Charlie Puth
Label: Artist Partner, Atlantic
Written By:
1. Charlie Puth
2. Jacob Kasher
Produced By:
1. Charlie Puth
Release Year: 2017
Price: Rs250

Dusk Till Dawn

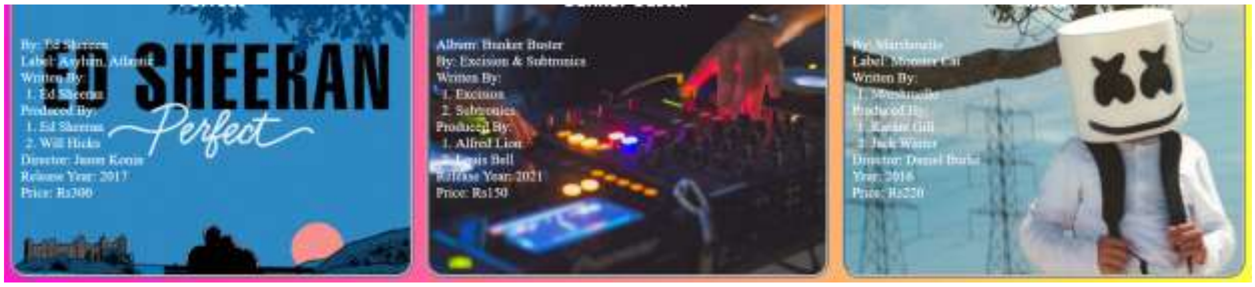
Album: Icarus Falls
By: Zayn Malik & Sia
Label: Republic
Written By:
1. Zayn Malik
2. Sia Furler
3. Alex Oriet
4. David Phelan
5. Greg Kurstin
Produced By:
1. Greg Kurstin
Director: Marc Webb
Release Year: 2018
Price: Rs[illegible]

Perfect

Bunker Buster

Alone

Figure 11: CSS styling3



By: Ed Sheeran
Label: Asylum, Atlantic
Written By:
1. Ed Sheeran
Produced By:
1. Ed Sheeran
2. Will Hicks
Director: Jason Koenig
Release Year: 2017
Price: Rs300

Album: Bunker Buster
By: Excision & Subtronics
Written By:
1. Excision
2. Subtronics
Produced By:
1. Alfred Lion
2. Louis Bell
Release Year: 2021
Price: Rs150

By: Marshmello
Label: Monstercat
Written By:
1. Marshmello
Produced By:
1. Karanr Gill
2. Jack Winter
Director: Daniel Burke
Year: 2016
Price: Rs220

Figure 12: CSS styling4

7. Difference between DTD and Schema

1. Document Type Definition (DTD):

DTD stands for Document Type Definition which is a document that defines the structure of an XML document. It specifies how to document elements are arranged and nested, as well as what elements are included in the documents and their attributes (Indika, 2011) . It can be classified into two types which are internal DTD and external DTD. In internal DTD it is specified inside a document whereas in external DTD it is specified outside a document. The main purpose of the DTD is to check the grammar and validity of an XML document which determines whether an XML document has a valid structure.

2. XML Schema Definition (XSD):

XSD stands for XML Schema Definition which is a way to describe the structure of an XML document. It specifies the rules for all the XML document's attributes and components and also specifies the elements are child elements and the order in which they should appear (Indika, 2011). XML schemas are commonly used in web applications meanwhile it is extensible and supports data types and namespaces. The greatest asset of schema is the ability to support data types. Moreover, the XML schema includes features for working with data in databases and converting between data types.

Table 7: Difference between DTD and XML Schema

S.NO.	DTD	XSD
1.	The declarations that describe a Standard Generalization Markup Language (SGML) document type are known as DTD.	The elements of an XML document are defined by the XSD.
2.	A namespace is not supported in DTD.	A namespace is supported in XSD.
3.	The order of child elements is not specified in the DTD.	The order of child elements is specified in the XSD.
4.	DTD is not extensible.	XSD is extensible.
5.	It doesn't support datatypes.	It supports data types for elements and attributes.

6.	It is more difficult than XSD in comparison.	It is relatively easier to understand than DTD.
7.	It doesn't give us much control over the XML document's structure.	It gives us more control over the XML document's structure.

(GeeksForGeek, 2020)

8. Coursework development

For this coursework, we must make a model for the Music store which provides a variety of songs with a wide range of genres. So as an XML developer XML document is made where for the validation Schema and DTD document are created which defines the XML also to render and design the document CSS file is created.

At first Tree diagram model of the XML document was made with the help of draw.io. Draw.io is a free online diagram software where we can create different kinds of model diagrams that are free to use. I mainly use this tool to draw different kinds of models and diagrams. From the tree diagram, we can list out all the elements and the attributes which we going to use while developing XML. This tree diagram contains the root element and its corresponding child elements.

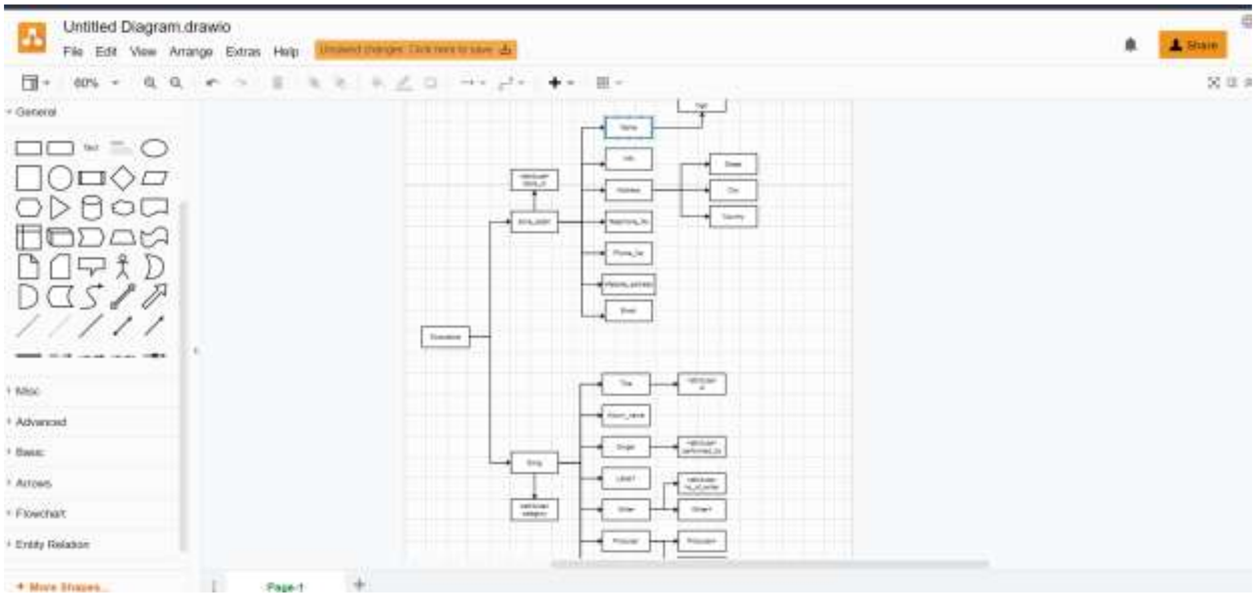
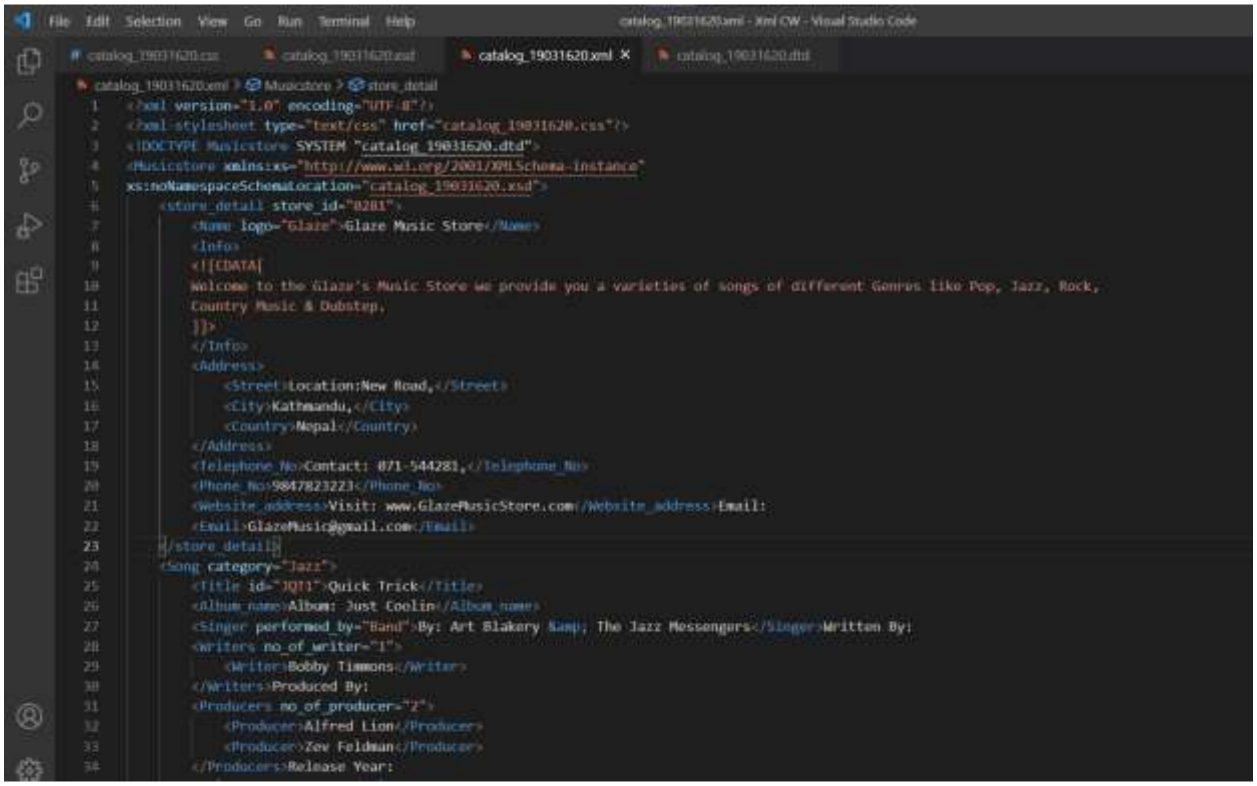


Figure 13: SS of draw.io

For the coding of the XML, Schema, and DTD, Visual Studio Code (VSE) editor is used. It is a very powerful, lightweight code editor which is mainly used in building and debugging modern web and cloud applications. It provides tons of features where we can debug our code easily with the help of extensions. It was a lot easier to find out the errors while validating XSD and DTD and was able to debug using the hints error messages in Visual Studio Code.

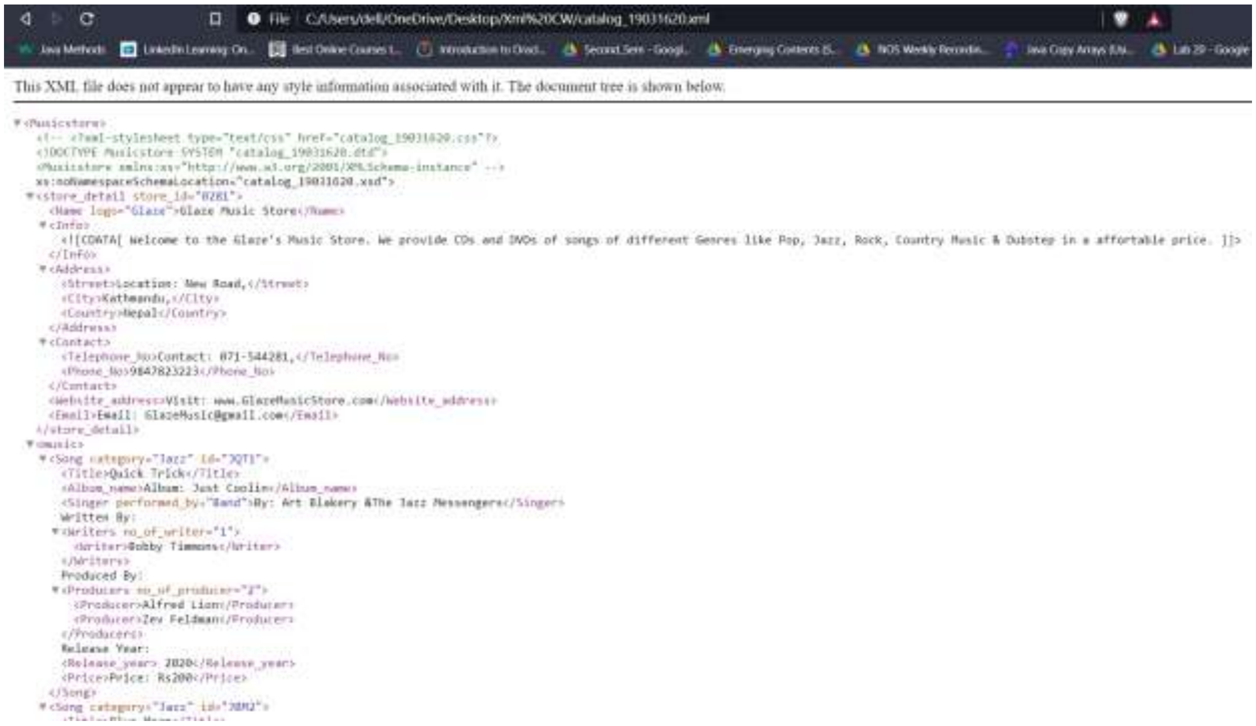

File Edit Selection View Go Run Terminal Help
catalog_19031620.xml - Xml CW - Visual Studio Code
catalog_19031620.css
catalog_19031620.xsd
catalog_19031620.xml
catalog_19031620.dtd
catalog_19031620.xml > Musicstore > store_detail

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="catalog_19031620.css"?>
<!DOCTYPE Musicstore SYSTEM "catalog_19031620.dtd">
<Musicstore xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
xs:noNamespaceSchemaLocation="catalog_19031620.xsd">
    <store_detail store_id="0281">
        <Name logo="Glaze">Glaze Music Store</Name>
        <Info>
        <![CDATA[
        Welcome to the Glaze's Music Store we provide you a varieties of songs of different Genres like Pop, Jazz, Rock,
        Country Music & Dubstep.
        ]]>
        </Info>
        <Address>
            <Street>Location:New Road,</Street>
            <City>Kathmandu,</City>
            <Country>Nepal</Country>
        </Address>
        <Telephone_No>Contact: 071-544281,</Telephone_No>
        <Phone_No>9847823223</Phone_No>
        <Website_address>Visit: www.GlazeMusicStore.com</Website_address>Email:
        <Email>GlazeMusic@gmail.com</Email>
    </store_detail>
    <Song category="Jazz">
        <Title id="JQT1">Quick Trick</Title>
        <Album_name>Album: Just Coolin</Album_name>
        <Singer performed_by="Band">By: Art Blakery &amp; The Jazz Messengers</Singer>Written By:
        <Writers no_of_writer="1">
            <Writer>Bobby Timmons</Writer>
        </Writers>Produced By:
        <Producers no_of_producer="2">
            <Producer>Alfred Lion</Producer>
            <Producer>Zev Feldman</Producer>
        </Producers>Release Year:
```

Figure 14: SS of using Visual Studio

At first XML document was made which stores the data of all songs. I created 15 song element which stores 15 data of songs. I implemented data for 2 jazz songs, 4 rock songs, 3 country music songs, 4 Pop songs, and 2 dubstep songs. I made the "Musicstore" element which has two child elements "store_detail" and "music" where store_detail contains the details of the store like name, address, contact, email, etc. whereas music contains the song element which has all the details of the songs like the title, album name, singer, writer, producer, etc.

I used Brave Browser to display the contents of the document. The browser helps to render the contents and displays the document. The figure below is the structure of the XML without the CSS.



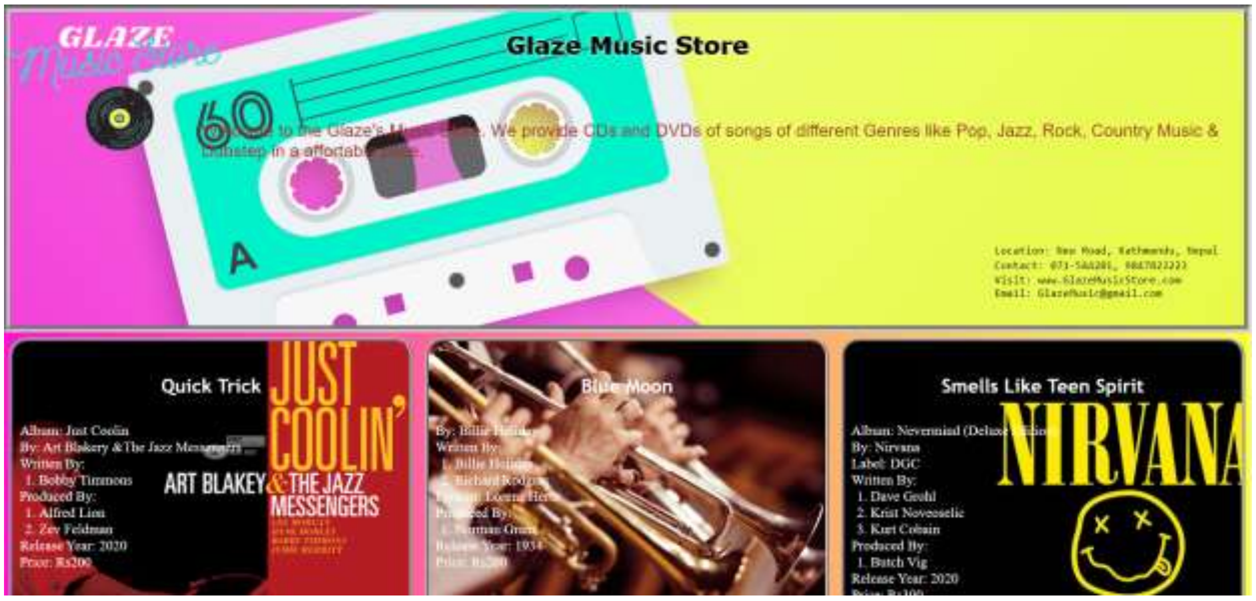
C:/Users/dell/OneDrive/Desktop/Xml%20CW/catalog_19031620.xml

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<Musicstore>
   <!-- <?xml-stylesheet type="text/css" href="catalog_19031620.css"?>
   <!DOCTYPE Musicstore SYSTEM "catalog_19031620.dtd">
   <Musicstore xmlns:xs="http://www.w3.org/2001/XMLSchema-instance" -->
   xs:noNamespaceSchemaLocation="catalog_19031620.xsd">
 ▼<store_detail store_id="0281">
    <Name logo="Glaze">Glaze Music Store</Name>
  ▼<Info>
     <![CDATA[ Welcome to the Glaze's Music Store. We provide CDs and DVDs of songs of different Genres like Pop, Jazz, Rock, Country Music & Dubstep in a affortable price. ]]>
    </Info>
  ▼<Address>
     <Street>Location: New Road,</Street>
     <City>Kathmandu,</City>
     <Country>Nepal</Country>
    </Address>
  ▼<Contact>
     <Telephone_No>Contact: 071-544281,</Telephone_No>
     <Phone_No>9847823223</Phone_No>
    </Contact>
    <Website_address>Visit: www.GlazeMusicStore.com</Website_address>
    <Email>Email: GlazeMusic@gmail.com</Email>
  </store_detail>
 ▼<music>
  ▼<Song category="Jazz" id="JQT1">
     <Title>Quick Trick</Title>
     <Album_name>Album: Just Coolin</Album_name>
     <Singer performed_by="Band">By: Art Blakery &The Jazz Messengers</Singer>
     Written By:
   ▼<Writers no_of_writer="1">
      <Writer>Bobby Timmons</Writer>
     </Writers>
     Produced By:
   ▼<Producers no_of_producer="2">
      <Producer>Alfred Lion</Producer>
      <Producer>Zev Feldman</Producer>
     </Producers>
     Release Year:
     <Release_year> 2020</Release_year>
     <Price>Price: Rs200</Price>
    </Song>
  ▼<Song category="Jazz" id="JBM2">
```

Figure 15: Displaying XML document without linking CSS

To add style and display the information CSS is used. The figure below is the XML document after applying CSS. With the help of the CSS, an entire XML document is formatted and displayed according to our wish.



Glaze Music Store

[illegible] to the Glaze's [illegible]. We provide CDs and DVDs of songs of different Genres like Pop, Jazz, Rock, Country Music & Dubstep in a affortable [illegible]

```
Location: New Road, Kathmandu, Nepal
Contact: 071-544281, 9847823223
Visit: www.GlazeMusicStore.com
Email: GlazeMusic@gmail.com
```

Quick Trick

Album: Just Coolin
By: Art Blakery &The Jazz Messengers
Written By:
1. Bobby Timmons
Produced By:
1. Alfred Lion
2. Zev Feldman
Release Year: 2020
Price: Rs200

Blue Moon

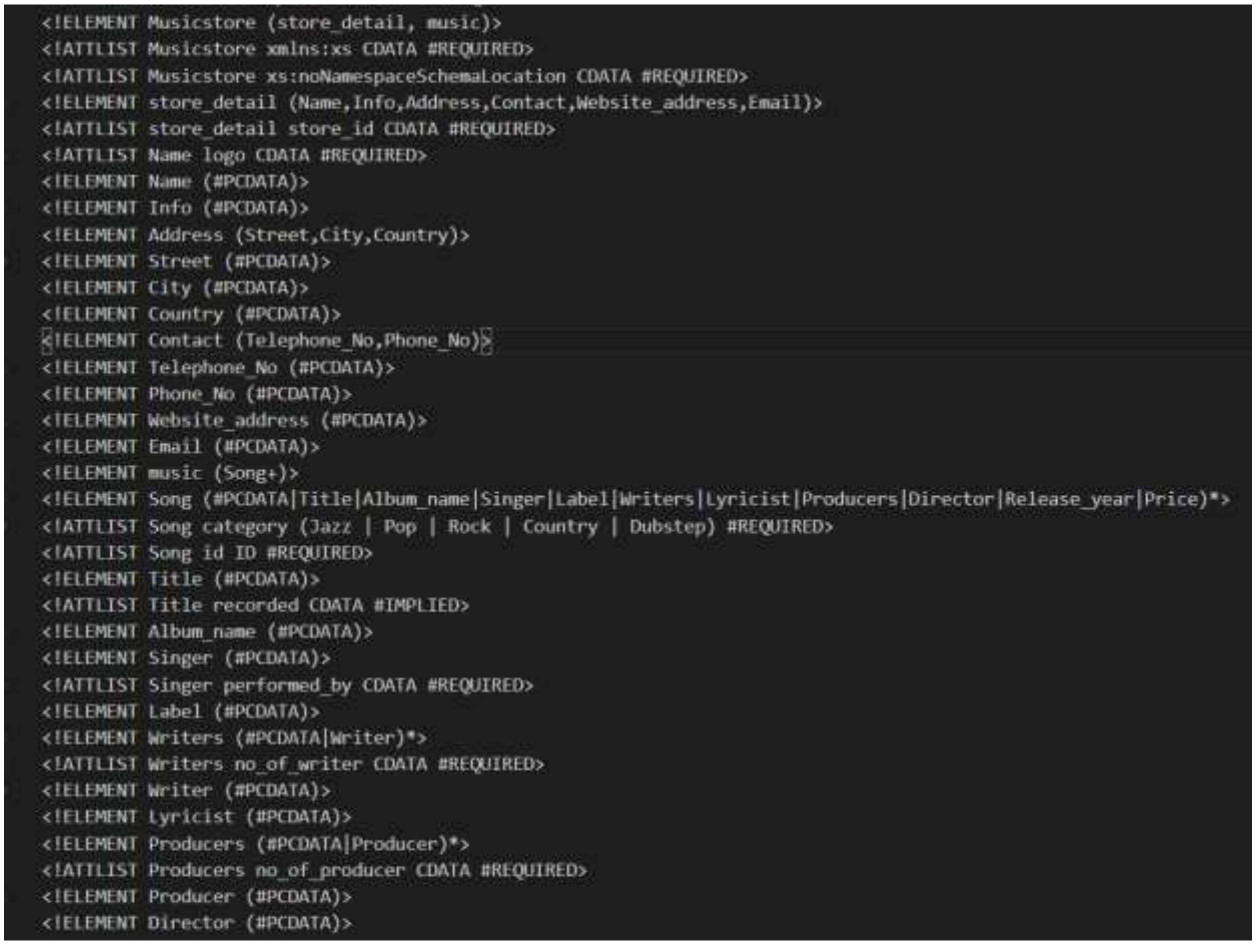
By: Billie Holiday
Written By:
1. Billie Holiday
2. Richard Rodgers
[illegible]
Produced By:
1. [illegible]
Release Year: 1934
Price: Rs200

Smells Like Teen Spirit

Album: Nevermind (Deluxe Edition)
By: Nirvana
Label: DGC
Written By:
1. Dave Grohl
2. Krist Novoselic
3. Kurt Cobain
Produced By:
1. Butch Vig
Release Year: 2020

Figure 16: Displaying XML after linking CSS

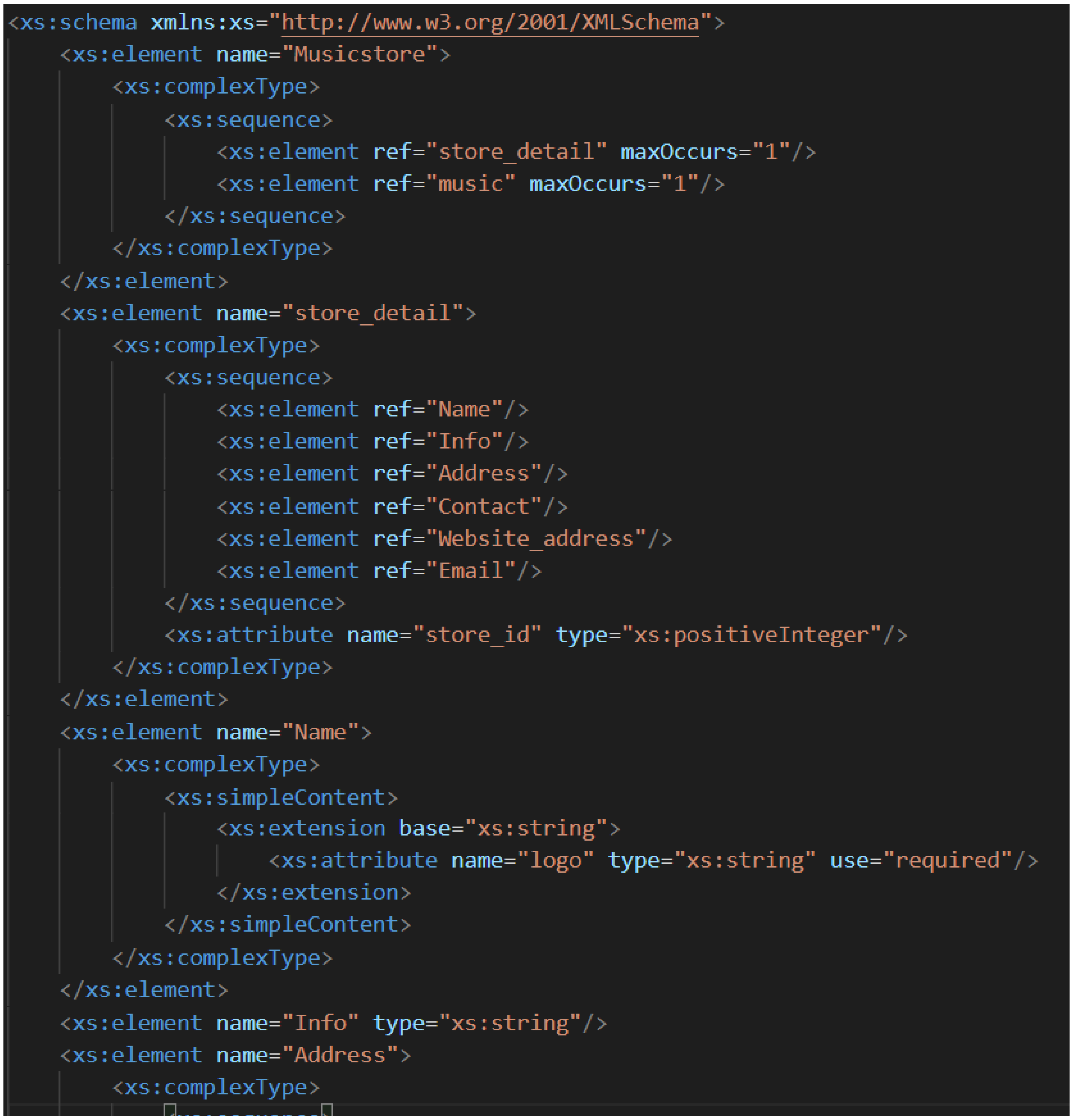
To define the XML and validate the contents of the XML, DTD and Schema are used as per the coursework. External DTD is used where it defined the XML document's structure, legal elements, and attributes.



```
<!ELEMENT Musicstore (store_detail, music)>
<!ATTLIST Musicstore xmlns:xs CDATA #REQUIRED>
<!ATTLIST Musicstore xs:noNamespaceSchemaLocation CDATA #REQUIRED>
<!ELEMENT store_detail (Name,Info,Address,Contact,Website_address,Email)>
<!ATTLIST store_detail store_id CDATA #REQUIRED>
<!ATTLIST Name logo CDATA #REQUIRED>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Info (#PCDATA)>
<!ELEMENT Address (Street,City,Country)>
<!ELEMENT Street (#PCDATA)>
<!ELEMENT City (#PCDATA)>
<!ELEMENT Country (#PCDATA)>
<!ELEMENT Contact (Telephone_No,Phone_No)>
<!ELEMENT Telephone_No (#PCDATA)>
<!ELEMENT Phone_No (#PCDATA)>
<!ELEMENT Website_address (#PCDATA)>
<!ELEMENT Email (#PCDATA)>
<!ELEMENT music (Song+)>
<!ELEMENT Song (#PCDATA|Title|Album_name|Singer|Label|Writers|Lyricist|Producers|Director|Release_year|Price)*>
<!ATTLIST Song category (Jazz | Pop | Rock | Country | Dubstep) #REQUIRED>
<!ATTLIST Song id ID #REQUIRED>
<!ELEMENT Title (#PCDATA)>
<!ATTLIST Title recorded CDATA #IMPLIED>
<!ELEMENT Album_name (#PCDATA)>
<!ELEMENT Singer (#PCDATA)>
<!ATTLIST Singer performed_by CDATA #REQUIRED>
<!ELEMENT Label (#PCDATA)>
<!ELEMENT Writers (#PCDATA|Writer)*>
<!ATTLIST Writers no_of_writer CDATA #REQUIRED>
<!ELEMENT Writer (#PCDATA)>
<!ELEMENT Lyricist (#PCDATA)>
<!ELEMENT Producers (#PCDATA|Producer)*>
<!ATTLIST Producers no_of_producer CDATA #REQUIRED>
<!ELEMENT Producer (#PCDATA)>
<!ELEMENT Director (#PCDATA)>
```

Figure 17: External DTD

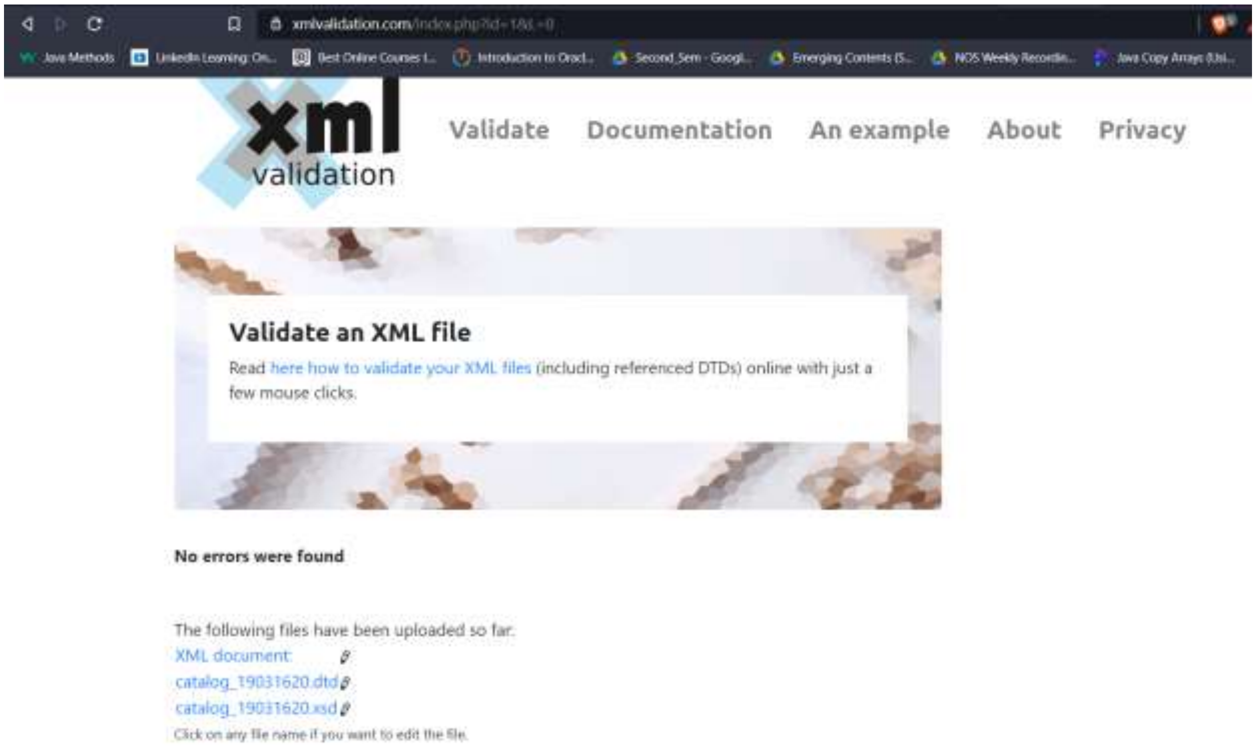
After that Schema also known as XSD is created which defines the legal building blocks of an XML document. There are different types of ways in which we can structure the Schemas among them I used the Flat Catalog Design where all the element declarations are made globally. The instance document's structure is built by referencing the global element declarations.



```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="Musicstore">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="store_detail" maxOccurs="1"/>
                <xs:element ref="music" maxOccurs="1"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="store_detail">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="Name"/>
                <xs:element ref="Info"/>
                <xs:element ref="Address"/>
                <xs:element ref="Contact"/>
                <xs:element ref="Website_address"/>
                <xs:element ref="Email"/>
            </xs:sequence>
            <xs:attribute name="store_id" type="xs:positiveInteger"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="Name">
        <xs:complexType>
            <xs:simpleContent>
                <xs:extension base="xs:string">
                    <xs:attribute name="logo" type="xs:string" use="required"/>
                </xs:extension>
            </xs:simpleContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="Info" type="xs:string"/>
    <xs:element name="Address">
        <xs:complexType>
```

Figure 18: Flat Catalog XSD

To validate and check the error www.xmlvalidation.com site was used through referencing DTD and Schema along with XML document. From this site, we were able to identify the errors and solve them according to them. Nevertheless, all the errors were successfully debugged and there was not an error on the XML document, DTD and XSD.



Validate an XML file

Read here how to validate your XML files (including referenced DTDs) online with just a few mouse clicks.

No errors were found

The following files have been uploaded so far:
XML document:
catalog_19031620.dtd
catalog_19031620.xsd
Click on any file name if you want to edit the file.

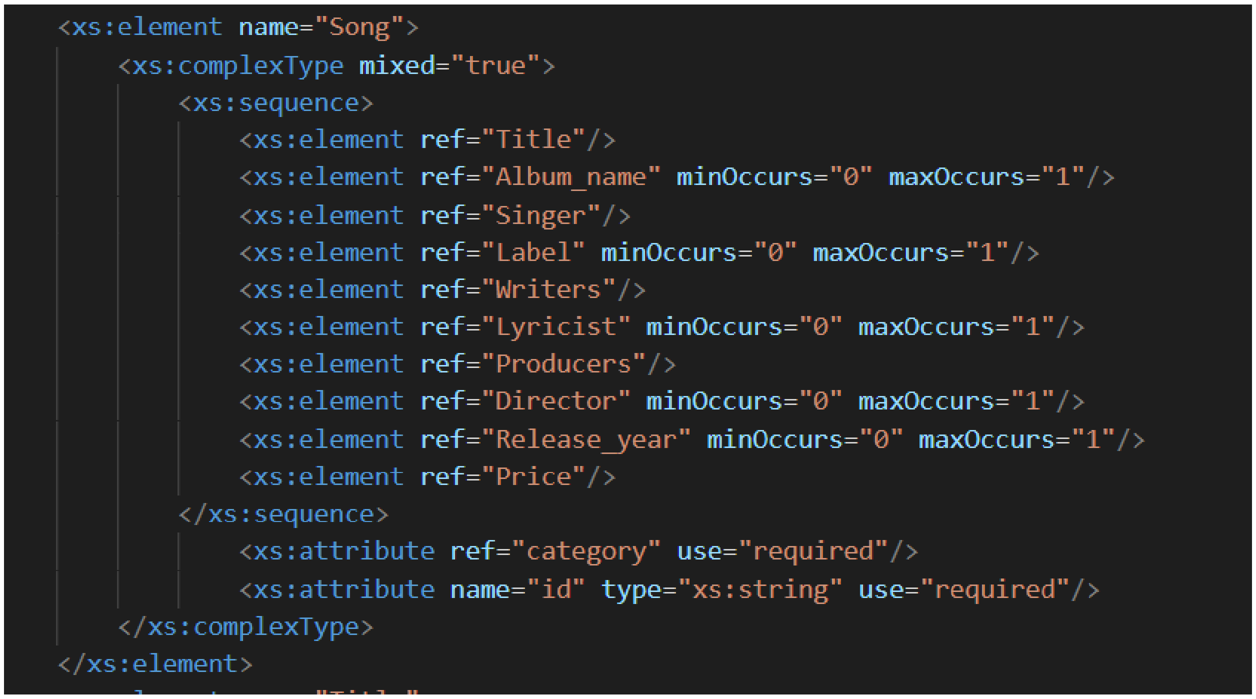
Figure 19: XML validation site

In this way, XML, DTD, and Schema along with CSS were developed which as a result a well-validated and well-formed XML document is created with proper design and rendering.

9.Critical Analysis

I gained a lot of things to know while developing the system for the Music Store. I was able to complete the project like an XML developer where DTD and Schemas are used to define the XML document along with the CSS for the design to render the contents in the browser. Some difficulties and problems were raised while developing this project as we were new and beginners in the XML. Several things were implemented in the XML document. I used 5 optional elements as per the coursework in the XML document which was described with the help of like Schema. implementing optional elements and attributes.

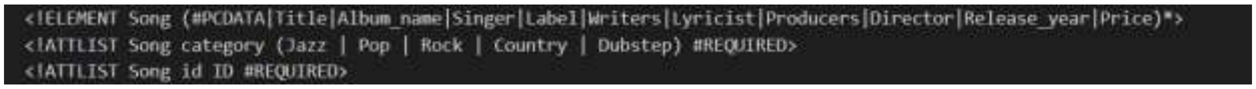
Here I used complex type with mixed content in the element Song and the element Album_name, Label, Lyricist, Director, and Release_year minOccurs is set 0 and maxOccurs is set to 1 which indicates that these elements are optional which are not necessarily required to be in the song element else only 1 element is allowed to be in the song element.



```
<xs:element name="Song">
    <xs:complexType mixed="true">
        <xs:sequence>
            <xs:element ref="Title"/>
            <xs:element ref="Album_name" minOccurs="0" maxOccurs="1"/>
            <xs:element ref="Singer"/>
            <xs:element ref="Label" minOccurs="0" maxOccurs="1"/>
            <xs:element ref="Writers"/>
            <xs:element ref="Lyricist" minOccurs="0" maxOccurs="1"/>
            <xs:element ref="Producers"/>
            <xs:element ref="Director" minOccurs="0" maxOccurs="1"/>
            <xs:element ref="Release_year" minOccurs="0" maxOccurs="1"/>
            <xs:element ref="Price"/>
        </xs:sequence>
            <xs:attribute ref="category" use="required"/>
            <xs:attribute name="id" type="xs:string" use="required"/>
    </xs:complexType>
</xs:element>
```

Figure 20: SS of XSD with a complex type of mixed content

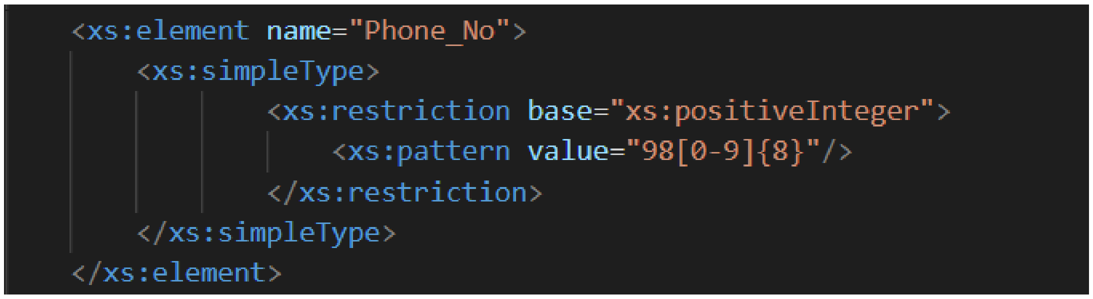
In the same way, I declared the mixed content in the DTD where the Song element can contain zero or more occurrence of the parsed character of data, “Title”, “Album_name”, “Singer”, “label”, “Writers”, “Lyricist”, “Producers”, “Director”, “Release_year” or “Price” elements.



```
<!ELEMENT Song (#PCDATA|Title|Album_name|Singer|Label|Writers|Lyricist|Producers|Director|Release_year|Price)*>
<!ATTLIST Song category (Jazz | Pop | Rock | Country | Dubstep) #REQUIRED>
<!ATTLIST Song id ID #REQUIRED>
```

Figure 21: SS of DTD with a complex type of mixed content

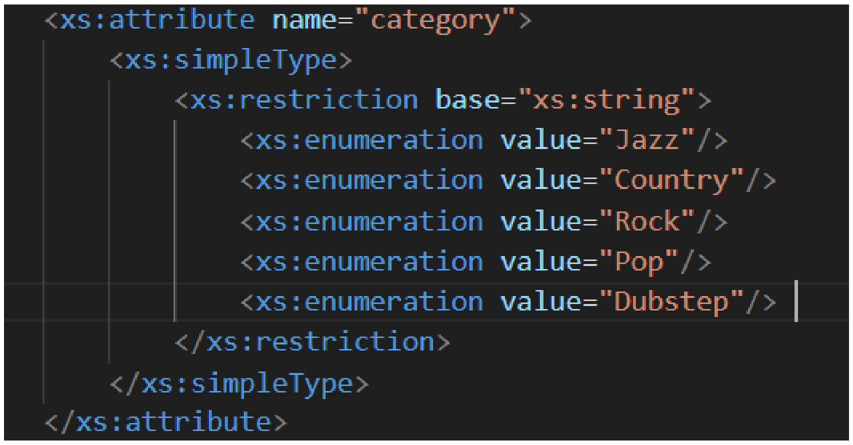
By using the Schema, I validated and restricted the data type I want to use like for example for a phone number where the first two numbers are 98 and the rest 8 numbers can be of any positive integer.



```
<xs:element name="Phone_No">
    <xs:simpleType>
            <xs:restriction base="xs:positiveInteger">
                <xs:pattern value="98[0-9]{8}"/>
            </xs:restriction>
    </xs:simpleType>
</xs:element>
```

Figure 22: SS of phone number pattern and restriction

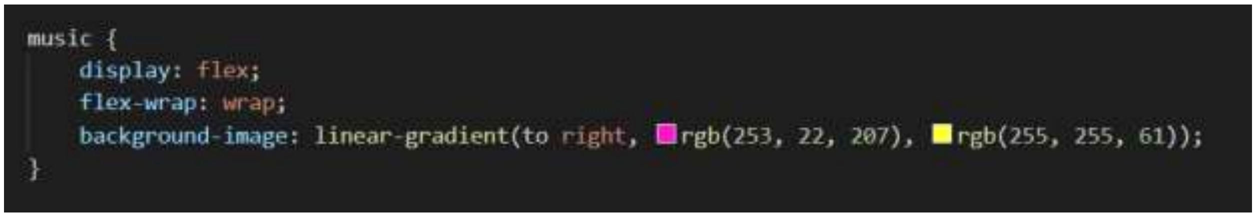
By the same, I also restricted and created the enumeration to follow the pattern in the attribute. For example, in the attribute of a song named category string restriction was made where only the Jazz, Country, Rock, Pop, and Dubstep values are allowed to be in the category attribute.



```
<xs:attribute name="category">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:enumeration value="Jazz"/>
            <xs:enumeration value="Country"/>
            <xs:enumeration value="Rock"/>
            <xs:enumeration value="Pop"/>
            <xs:enumeration value="Dubstep"/>
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
```

Figure 23: SS of enumeration in category attribute

I used many things in the CSS where I used the flex to display the contents of songs systematically. Also, linear-gradient color was used in the background.



```
music {
    display: flex;
    flex-wrap: wrap;
    background-image: linear-gradient(to right, rgb(253, 22, 207), rgb(255, 255, 61));
}
```

Figure 24: Showing flex display code

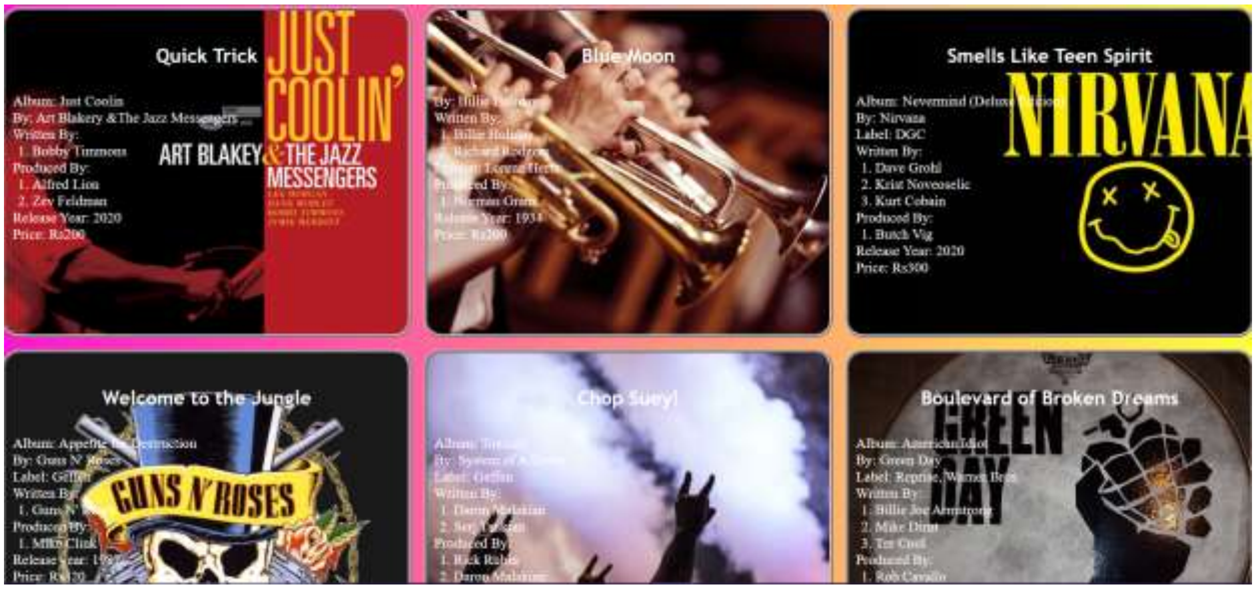
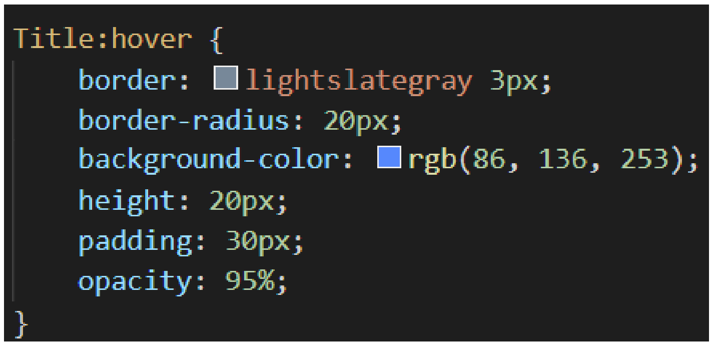


Figure 25: After using flex

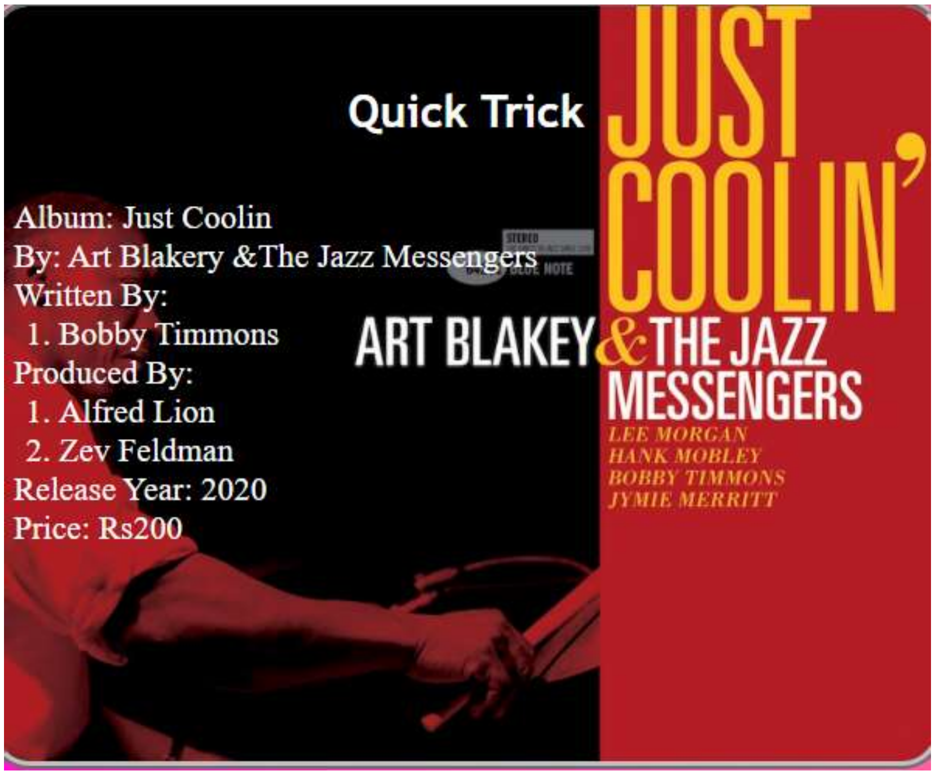
I placed the hover effect in the title of the song which changes the background color height and padding in the title.



```
Title:hover {
    border: lightslategray 3px;
    border-radius: 20px;
    background-color: rgb(86, 136, 253);
    height: 20px;
    padding: 30px;
    opacity: 95%;
}
```

Figure 26: Hover effect code

Here is the content of the song without the hover.

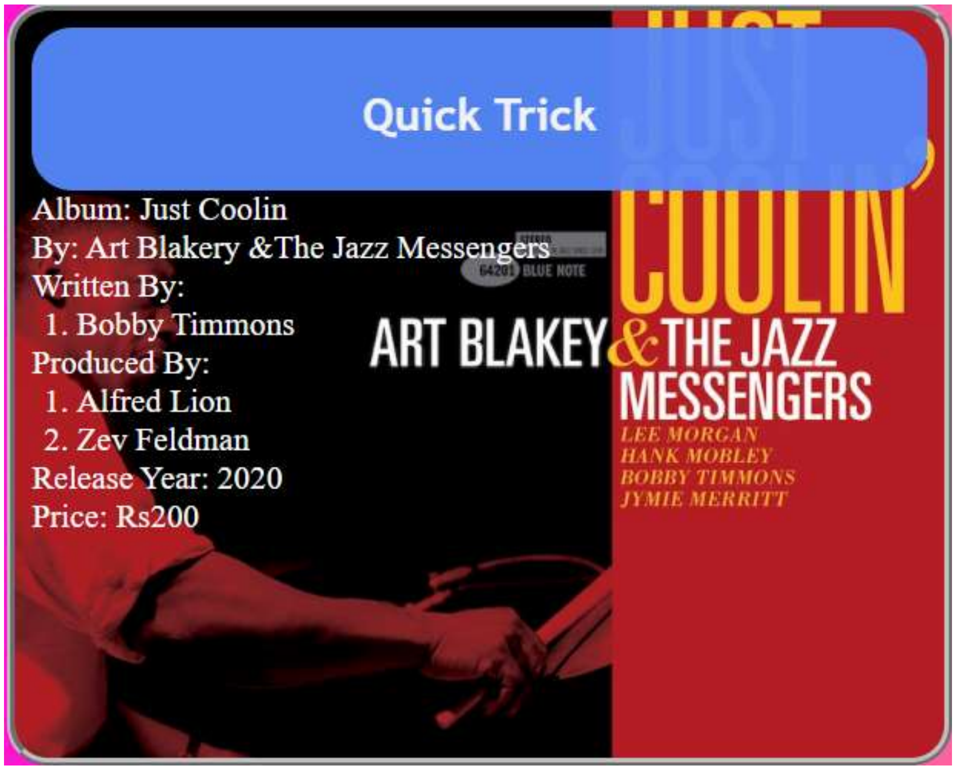


Quick Trick

Album: Just Coolin
By: Art Blakery &The Jazz Messengers
Written By:
1. Bobby Timmons
Produced By:
1. Alfred Lion
2. Zev Feldman
Release Year: 2020
Price: Rs200

Figure 27: Before hovering in Title

Here is the content of the song while placing the cursor over the title.

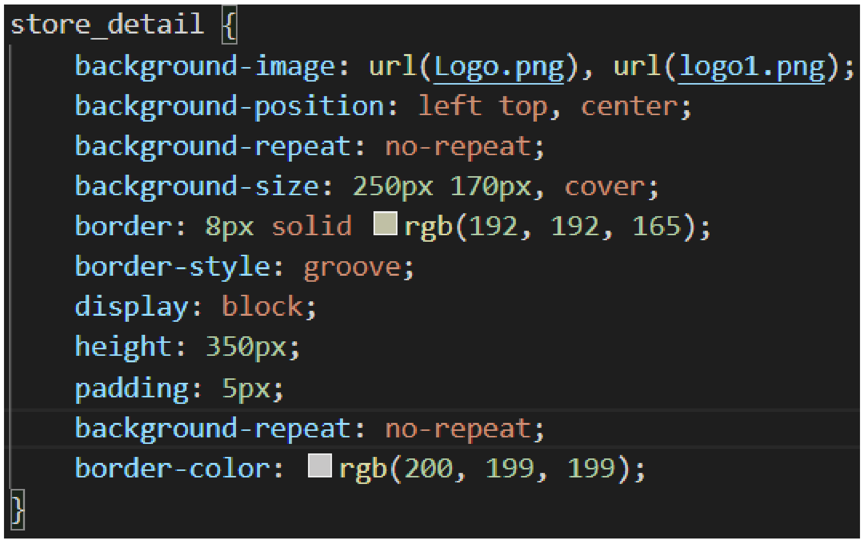


Quick Trick

Album: Just Coolin
By: Art Blakery &The Jazz Messengers
Written By:
1. Bobby Timmons
Produced By:
1. Alfred Lion
2. Zev Feldman
Release Year: 2020
Price: Rs200

Figure 28: After hovering in Title

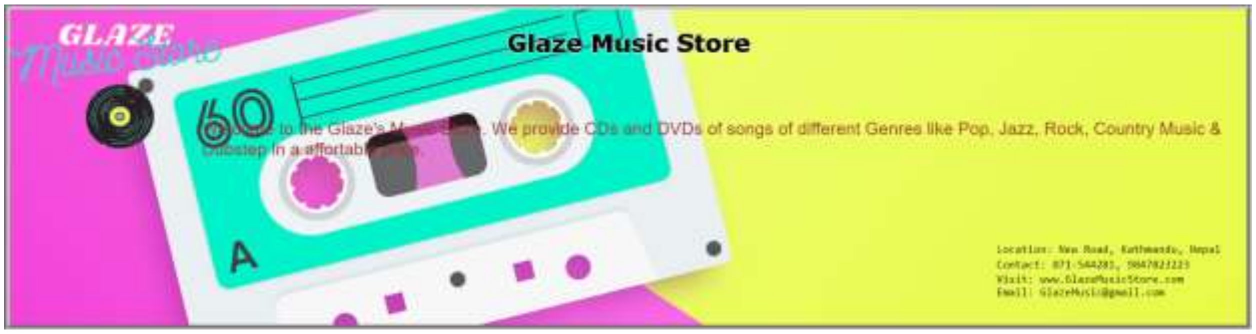
I was able to display the logo and the background image into the document by giving links of the image and logo in the background image using CSS.



```
store_detail {
    background-image: url(Logo.png), url(logo1.png);
    background-position: left top, center;
    background-repeat: no-repeat;
    background-size: 250px 170px, cover;
    border: 8px solid rgb(192, 192, 165);
    border-style: groove;
    display: block;
    height: 350px;
    padding: 5px;
    background-repeat: no-repeat;
    border-color: rgb(200, 199, 199);
}
```

Figure 29: Inserting logo in CSS

As a result, the logo and the background image were displayed.



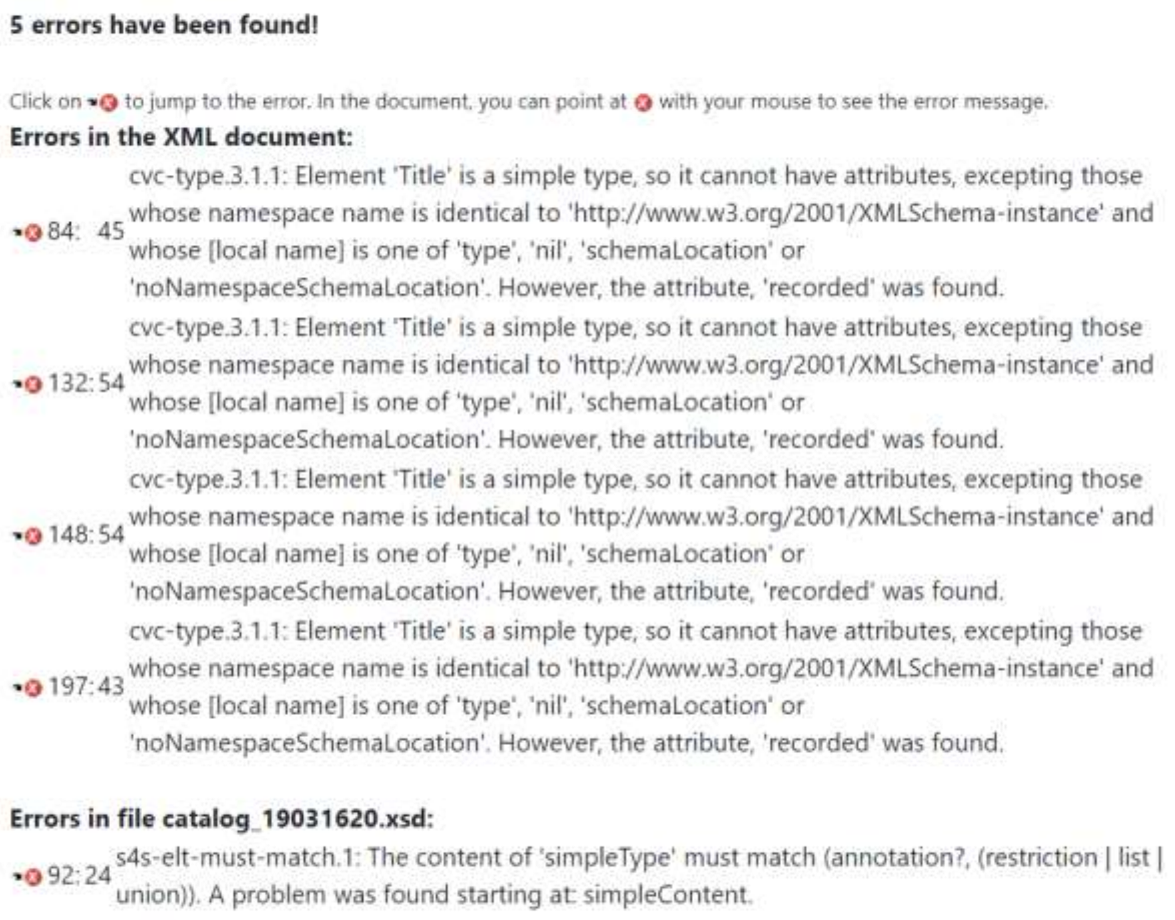
Glaze Music Store

[illegible] to the Glaze's [illegible]. We provide CDs and DVDs of songs of different Genres like Pop, Jazz, Rock, Country Music & Dubstep in a affortabl[illegible].

Location: New Road, Kathmandu, Nepal
Contact: 071-544281, 9847823223
Visit: www.GlazeMusicStore.com
Email: GlazeMusic@gmail.com

Figure 30: Displaying logo with background image

There were some problems I encountered while creating the XML document along with the DTD and Schema. The main problem I faced was on the Schema from which we had to bound the legal elements and give the elements data type restriction and patterns. It was a bit confusing while validating with Schema. Some of the errors I faced in the Schema are:



5 errors have been found!

Click on to jump to the error. In the document, you can point at with your mouse to see the error message.

Errors in the XML document:

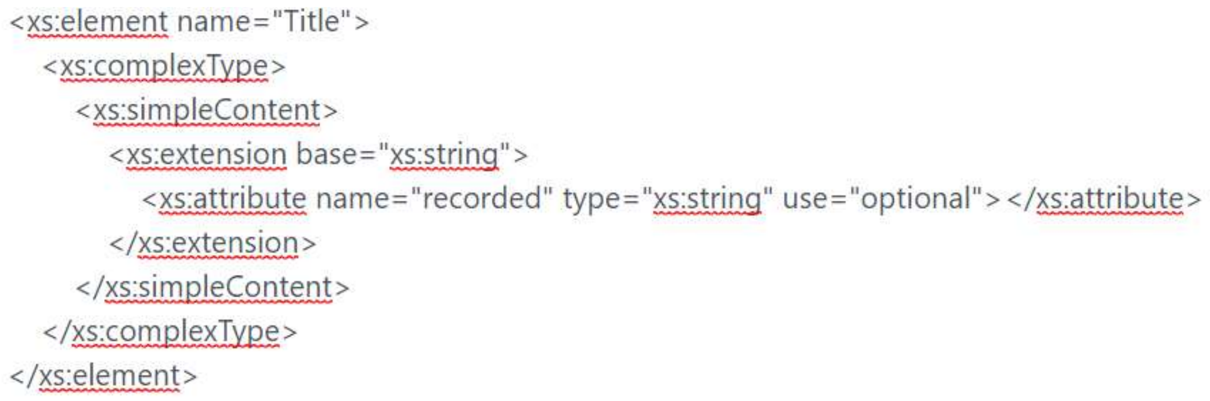
```
84:  45   cvc-type.3.1.1: Element 'Title' is a simple type, so it cannot have attributes, excepting those
          whose namespace name is identical to 'http://www.w3.org/2001/XMLSchema-instance' and
          whose [local name] is one of 'type', 'nil', 'schemaLocation' or
          'noNamespaceSchemaLocation'. However, the attribute, 'recorded' was found.
132:54    cvc-type.3.1.1: Element 'Title' is a simple type, so it cannot have attributes, excepting those
          whose namespace name is identical to 'http://www.w3.org/2001/XMLSchema-instance' and
          whose [local name] is one of 'type', 'nil', 'schemaLocation' or
          'noNamespaceSchemaLocation'. However, the attribute, 'recorded' was found.
148:54    cvc-type.3.1.1: Element 'Title' is a simple type, so it cannot have attributes, excepting those
          whose namespace name is identical to 'http://www.w3.org/2001/XMLSchema-instance' and
          whose [local name] is one of 'type', 'nil', 'schemaLocation' or
          'noNamespaceSchemaLocation'. However, the attribute, 'recorded' was found.
197:43    cvc-type.3.1.1: Element 'Title' is a simple type, so it cannot have attributes, excepting those
          whose namespace name is identical to 'http://www.w3.org/2001/XMLSchema-instance' and
          whose [local name] is one of 'type', 'nil', 'schemaLocation' or
          'noNamespaceSchemaLocation'. However, the attribute, 'recorded' was found.
```

Errors in file catalog_19031620.xsd:

```
92: 24  s4s-elt-must-match.1: The content of 'simpleType' must match (annotation?, (restriction | list |
        union)). A problem was found starting at: simpleContent.
```

Figure 31: Error in Schema

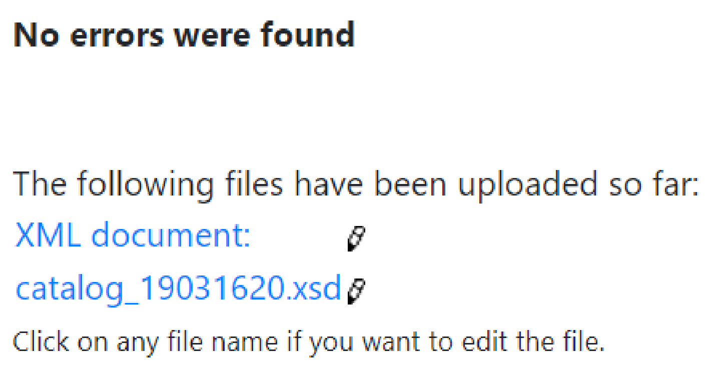
From this error, I found that the element "Title" was of simpleType but Title contains the attributes recorded so it must be a complexType. Therefore, I debugged the error replacing simpleType with the ComplexType in the element "Title" on the DTD document.



```
<xs:element name="Title">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="recorded" type="xs:string" use="optional"></xs:attribute>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

Figure 32: Debugging error in Schema

As a result, no errors were found.

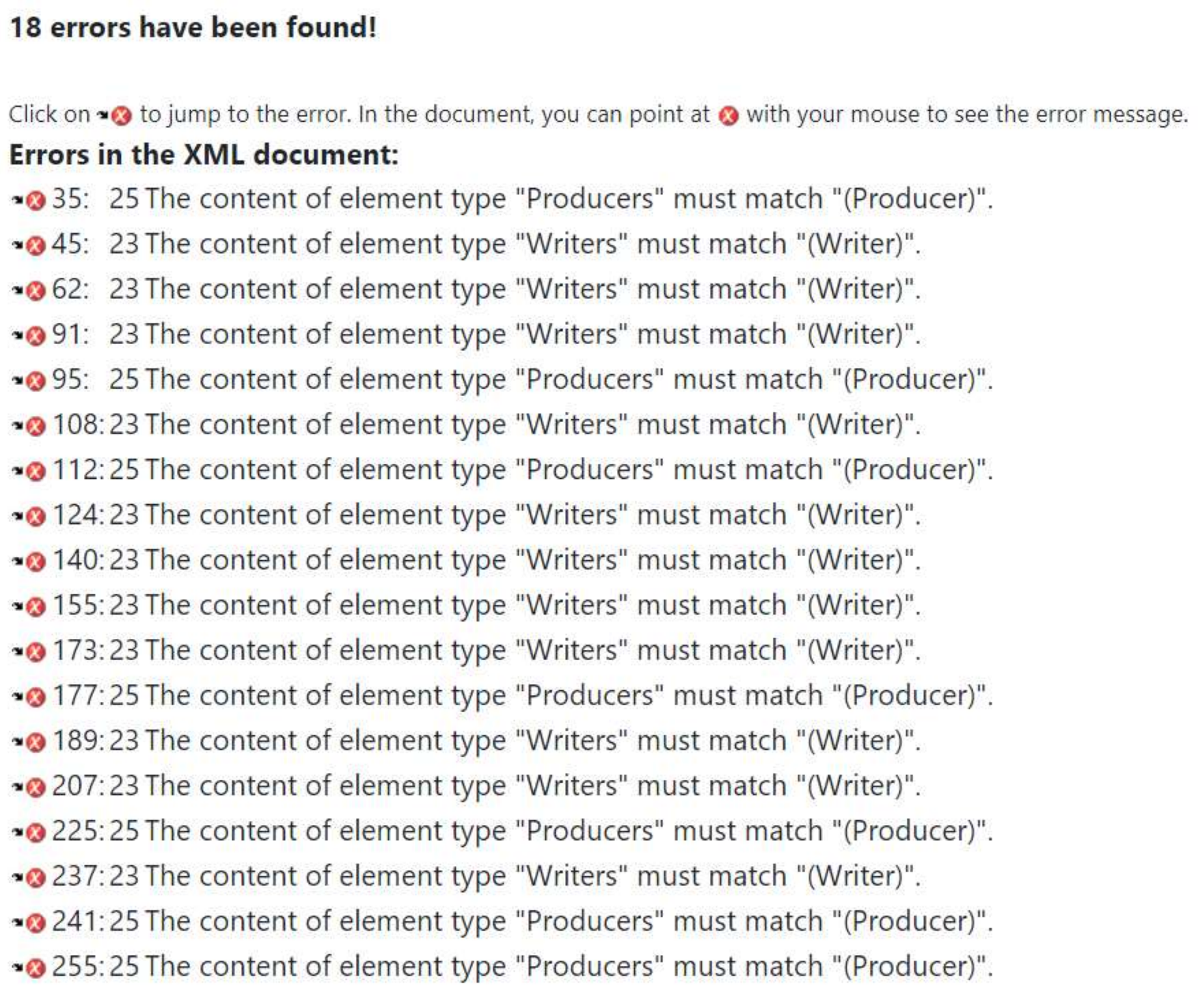


No errors were found

The following files have been uploaded so far:
XML document:
catalog_19031620.xsd
Click on any file name if you want to edit the file.

Figure 33: Validating after debugging

I also found errors in the DTD while validating like:



18 errors have been found!

Click on to jump to the error. In the document, you can point at with your mouse to see the error message.

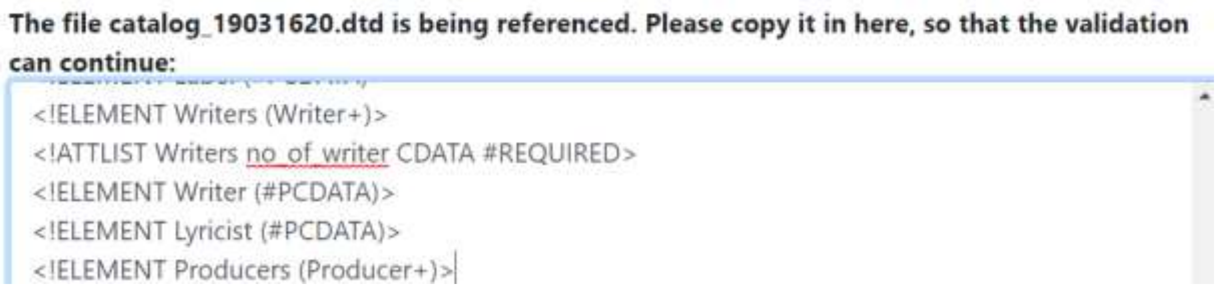
Errors in the XML document:

```
35:  25 The content of element type "Producers" must match "(Producer)".
45:  23 The content of element type "Writers" must match "(Writer)".
62:  23 The content of element type "Writers" must match "(Writer)".
91:  23 The content of element type "Writers" must match "(Writer)".
95:  25 The content of element type "Producers" must match "(Producer)".
108: 23 The content of element type "Writers" must match "(Writer)".
112: 25 The content of element type "Producers" must match "(Producer)".
124: 23 The content of element type "Writers" must match "(Writer)".
140: 23 The content of element type "Writers" must match "(Writer)".
155: 23 The content of element type "Writers" must match "(Writer)".
173: 23 The content of element type "Writers" must match "(Writer)".
177: 25 The content of element type "Producers" must match "(Producer)".
189: 23 The content of element type "Writers" must match "(Writer)".
207: 23 The content of element type "Writers" must match "(Writer)".
225: 25 The content of element type "Producers" must match "(Producer)".
237: 23 The content of element type "Writers" must match "(Writer)".
241: 25 The content of element type "Producers" must match "(Producer)".
255: 25 The content of element type "Producers" must match "(Producer)".
```

Figure 34: Error in DTD

From this error, I got to know that the child element of the element "Producer" and "Writers" are not matching. The element "Producer" and "Writers" had the child element "Producer" and "Writer" which occurs multiple times in the particular parent element.

Here the "+" was missing inside the element Writers and Producers so I included the "+" symbol in the element writer and producer which indicates that there are multiple child elements in the parent element Producers and Writers.

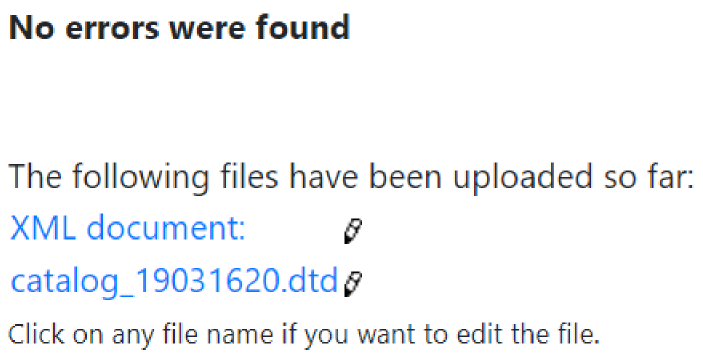


The file catalog_19031620.dtd is being referenced. Please copy it in here, so that the validation can continue:

```
<!ELEMENT Writers (Writer+)>
<!ATTLIST Writers no_of_writer CDATA #REQUIRED>
<!ELEMENT Writer (#PCDATA)>
<!ELEMENT Lyricist (#PCDATA)>
<!ELEMENT Producers (Producer+)>
```

Figure 35: Debugging error in DTD

In this way, an error was solved.



No errors were found

The following files have been uploaded so far:
XML document:
catalog_19031620.dtd
Click on any file name if you want to edit the file.

Figure 36: Validating after debugging

By solving these errors while I encountered and creating different types of validation, restriction, data type with the proper CSS I was able to complete the project of the Music store by XML.

10. Conclusion

For this coursework, I made a model for the Music store named Glaze Music Store which provides a variety of songs with a wide range of genres like jazz, rock, pop, country, and dubstep. So as an XML developer XML document was made where for the validation Schema and DTD document were created which defined the XML also to render and design the document CSS file was created. A tree diagram was made to list out all the elements and attributes which was used while developing the XML document. The root elements with their child elements were shown in the tree diagram. Testing was done to check the well-form and validation of the XML and also the CSS was checked whether the XML is rendered according to it or not. With the help of the www.xmlvalidation.com site, we were able to find and check the errors of the DTD and Schema along with the XML document. Therefore, in the end, after debugging all the errors testing was completed successfully.

From this coursework I was able to gain an understanding of the XML and how does it work. I got to know why the validation in the XML was important and why we needed the DTD and Schema. I got to know how to the XML with the help of CSS. I was again to gain the concepts on how and when to use the patterns, restriction, and data type in the Schema to validate and define the legal building blocks of an XML document. I was able to do a real work project just like an XML developer from this coursework. I faced some difficulties and problems while doing this project but with the help of the respective module tutor, lecture slides, and educational sites like w3schools I was able to debug the errors. As we practiced and did our lab work in the tutorial and workshop class it was quite easy for me to develop the XML and DTD. But I faced some difficulties with the Schema for which a lot of research was done. I was confused with some questions of the coursework but with the help of the module teacher and friends, those queries were solved. In this way, I was able to complete the coursework on time.

11. References

ASQ, 2021. *WHAT IS A TREE DIAGRAM?.* [Online] Available at: https://asq.org/quality-resources/tree-diagram
[Accessed 6 5 2021].

GeeksForGeek, 2020. *Difference between Document Type Definition (DTD) and XML Schema Definition (XSD).* [Online] Available at: https://www.geeksforgeeks.org/difference-between-document-type-definition-dtd-and-xml-schema-definition-xsd/
[Accessed 6 5 2021].

Indika, 2011. *Difference Between XML Schema and DTD.* [Online] Available at: https://www.differencebetween.com/difference-between-xml-schema-and-vs-dtd/
[Accessed 27 4 2021].